PROYECTO DE TITULACIÓN

CARRERA

Ingeniería en Tecnologías de la Información

TEMA

Diseño e Implementación de un Sistema de Control de Ingreso basado en IoT y Blockchain

AUTORES

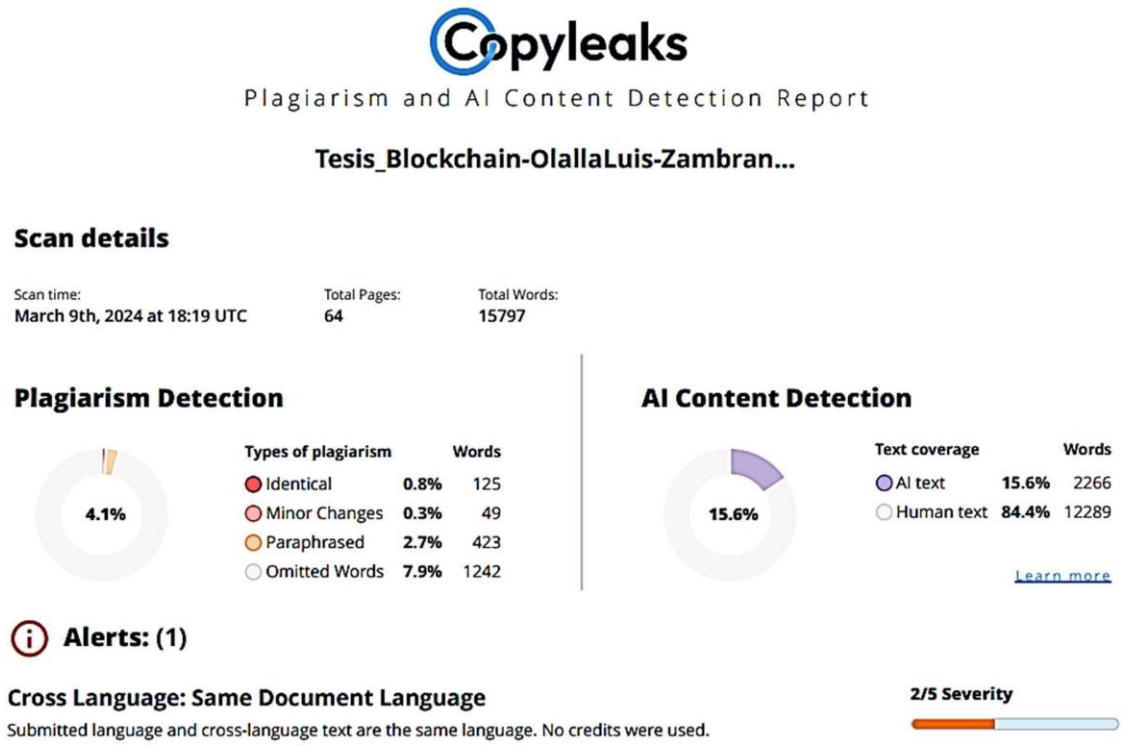
Olalla Chuque, Luis Miguel y Zambrano Macias, Jhon Kleber

TUTOR

Ing. Rodríguez Galán, German Eduardo, Mgtr.

Santo Domingo, 01 de marzo de 2024

Reporte de Verificación de Contenido



Firma:

Firmado electrónicamente por:
GERMAN EDUARDO
RODRIGUEZ GALAN

.....

Ing. Rodríguez Galán, Germán Eduardo Mgtr.

C. C. 0603431685

Departamento de Ciencias de la Computación

Carrera de Ingeniería en Tecnologías de la Información

Certificación

Certifico que el trabajo de integración curricular: "**Diseño e Implementación de un Sistema de Control de Ingreso basado en IoT y Blockchain**" fue realizado por los señores **Olalla Chuque, Luis Miguel** y **Jhon Kleber, Zambrano Macias**, el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizada en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Santo Domingo, 01 de marzo de 2024

Firma:

Firmado electrónicamente por:
GERMAN EDUARDO
RODRIGUEZ GALAN

.....

Ing. Rodríguez Galán, Germán Eduardo Mgtr.

C. C. 0603431685

Departamento de Ciencias de la Computación

Carrera de Ingeniería en Tecnologías de la Información

Responsabilidad de Autoría

Nosotros, **Luis Miguel, Olalla Chuque**, con cédula de ciudadanía N.° **2350439861** y **Jhon Kleber, Zambrano Macias** con cédula de ciudadanía N.° **2350051930 ,** declaramos que el contenido, ideas y criterios del trabajo de integración curricular: **Diseño e Implementación de un Sistema de Control de Ingreso basado en IoT y Blockchain** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Santo Domingo, 01 de marzo de 2024

Firma

Olalla Chuque, Luis Miguel

C.C.: 2350439861

Firma

Zambrano Macias, Jhon Kleber

C.C.: 2350051930

Departamento de Ciencias de la Computación

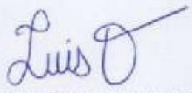
Carrera de Ingeniería en Tecnologías de la Información

Autorización de Publicación

Nosotros, **Olalla Chuque, Luis Miguel**, con cédula de ciudadanía N.° **2350439861** y **Zambrano Macias, Jhon Kleber**, con cédula de ciudadanía N.° **2350051930,** autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de integración curricular: **Diseño e Implementación de un Sistema de Control de Ingreso basado en IoT y Blockchain** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Santo Domingo, 01 de marzo de 2024

Firma



Olalla Chuque, Luis Miguel

C.C.: 2350439861

Firma

Zambrano Macias, Jhon Kleber

C.C.: 2350051930

Dedicatoria

Dedico esta tesis a mis padres, Iván Olalla y Aracely Chuque, quienes me han ayudado en todo el transcurso de mi vida y de la carrera universitaria, por haberme enseñado a ser una persona de bien. Dedico esta tesis también a mis amigos Genesis Heredia, Carlos Lucio y a mi compañero de tesis y amigo Jhon Zambrano, quienes han sido un gran apoyo en muchos aspectos.

Luis Miguel Olalla Chuque

Dedicatoria

A mis amados padres, Leticia Macias y Kleber Zambrano quienes han sido la brújula que guía cada paso de este camino académico. A mi querida hermana, Dayana Zambrano por ser mi compañera de aventuras, mi confidente y mi más grande admiradora. A todos aquellos que, a lo largo de mi vida universitaria, me han brindado su amistad, su apoyo y su sabiduría. A aquellos que me han llamado amigo, compañero o simplemente me han tendido una mano en los momentos difíciles. Finalmente, a mi compañero y amigo de tesis Luis Olalla quien con esfuerzo y dedicación logramos cumplir nuestros objetivos planteados.

Jhon Kleber Zambrano Macias

Agradecimiento

Al terminar nuestro trabajo de integración curricular, así como toda la etapa universitaria, queremos agradecer a nuestro docente a cargo el Ing. Germán Rodríguez. Quien ha sido de mucha ayuda partiendo sus conocimientos a nosotros los estudiantes. Le estamos completamente agradecidos por hacer posible la culminación de este proyecto. Queremos agradecer a cada docente que a lo largo de nuestra carrera nos dio las bases necesarias para poder ir aprendiendo y mejorando cada día. Por último, gracias a la Universidad de las Fuerzas Armadas ESPE al ser una institución de alto prestigio habernos brindado la oportunidad de cursar y finalizar nuestra carrera universitaria.

Olalla Luis y Jhon Zambrano

Índice de contenido

Índice de Tablas

Índice de Figuras

Resumen

Este estudio presenta la implementación de un sistema innovador de control de acceso basado en la Internet de las Cosas (IoT) y la tecnología de Blockchain. Esta implementación se centra en mejorar la seguridad en el control de ingreso a instalaciones, ofreciendo una solución que combina la conectividad de dispositivos IoT (Raspberry Pi) asegurando la información a través de cadenas de bloques (Blockchain). El sistema propuesto integra un control de acceso mediante Identificación por Radio Frecuencia (RFID) y dispositivos IoT para recopilar datos en tiempo real sobre las interacciones de los usuarios con el sistema de control de ingreso. Este sistema representa una solución prometedora para aplicaciones de seguridad en entornos críticos, como edificios gubernamentales, instalaciones militares y empresas de alto valor. Además, su potencial se extiende a otros campos donde la integridad de los registros y la autenticación de usuarios son esenciales. Este trabajo contribuye al crecimiento de la seguridad en la era digital, destacando la importancia de la combinación de IoT y Blockchain en aplicaciones de control de acceso.

Palabras clave: Raspberry Pi, Blockchain, Internet de las cosas.

Abstract

This study presents the implementation of an innovative access control system based on the Internet of Things (IoT) and Blockchain technology. This implementation is focused on improving security in facility access control, offering a solution that combines the connectivity of IoT devices (raspberry pi) securing information through blockchain. The proposed system integrates access sensors and IoT devices to collect real-time data on user interactions with the entrance control system. This system represents a promising solution for security applications in critical environments, such as government buildings, military installations, and high-value enterprises. In addition, its potential extends to other fields where log integrity and user authentication are essential. This work contributes to the growth of security in the digital age, highlighting the importance of combining IoT and Blockchain in access control applications.

Keywords: Raspberry Pi, Blockchain, Internet of Things.

1. Introducción

El presente trabajo se fundamenta en la necesidad de mejorar el proceso de control de acceso que se ejecuta en la Universidad de las Fuerzas Armadas sede Santo Domingo. En la actualidad, este proceso se ejecuta de forma manual, los señores guardias son responsables de registrar los vehículos que ingresan y salen, así como identificar al personal civil y militar. Por otro lado, los estudiantes no tienen un control de ingreso debido al tiempo que representa el registro manual de todos y cada uno.

La ausencia de tecnologías automatizadas para la gestión de este control de ingreso de personal a la sede ha generado la inquietud de explorar soluciones más eficientes y avanzadas. La dependencia de métodos manuales nos ha planteado un desafío en términos de interconectividad y seguridad.

La revisión de la literatura revela que, en entornos similares, se han implementado exitosamente tecnologías modernas, como sistemas de control de acceso basados en la integración de dispositivos electrónicos y software especializado. Estas soluciones no solo agilizan el proceso, sino que también mejoran la precisión y la seguridad, proporcionando una base sólida para la gestión eficiente de los accesos.

El acceso eficiente y seguro a las instituciones educativas es un aspecto importante para garantizar un ambiente adecuado para el aprendizaje. En el caso particular de la Universidad de las Fuerzas Armadas ESPE, sede Santo Domingo, el acceso de estudiantes, vehículos, personal civil y militar, se realiza a través de una única puerta de ingreso. En este único punto de control, uno o dos guardias son los encargados de facilitar el ingreso y realizar la apertura de la puerta para los vehículos que ingresan a la universidad.

Hasta el momento, no se ha incorporado ninguna tecnología para gestionar el control de ingreso de estudiantes, personal civil, militar, y vehículos en la sede de Santo Domingo. Por lo tanto, este estudio abordará estas limitaciones identificadas en este proceso, proponiendo la implementación de tecnologías innovadoras, específicamente aquellas relacionadas con la automatización del control de acceso mediante dispositivos IoT y Blockchain.

1.1 Estado del Arte

Se ejecutó una revisión de la literatura (SLR), siguiendo las directrices de Barbara Kitchenham y Brereton (2013). Este proceso incluyó la planificación, ejecución y documentación exhaustiva de la revisión con el objetivo de elaborar un estado del arte. Los

trabajos relacionados se centraron en la implementación de cadenas de bloques a través de dispositivos IoT.

Esta necesidad de revisión surgió ante la falta de estudios exhaustivos que aborden de manera integral y actualizada la implementación de un sistema de control de acceso integrando tecnología blockchain. Dada la creciente importancia de la protección de los datos, así como también las vulnerabilidades que existen actualmente, fue necesario recopilar y analizar las investigaciones existentes dentro de este campo con el objetivo de cubrir brechas en el conocimiento, ver una oportunidad de mejora de atributos como la eficiencia, calidad y transparencias en varios de los sistemas de control de acceso.

Por otro lado, hemos planteado la siguiente hipótesis: "*La implementación de un sistema de control de acceso que combine tecnologías blockchain e IoT en la Universidad de las Fuerzas Armadas ESPE, ubicada en Santo Domingo de los Tsáchilas, tiene el potencial de optimizar la gestión de ingreso de estudiantes al proporcionar mayor seguridad, eficiencia y transparencia en el proceso".*

Hemos identificado los términos clave de nuestra propuesta: ***IoT, Blockchain y Raspberry Pi***, con los que hemos construido nuestra cadena principal para buscar trabajos relacionados en los diferentes repositorios de documentos con información relacionada. Se han incluido, dentro de nuestro análisis, los trabajos y propuestas de los últimos 5 años, escritas en español, que hayan presentado una metodología clara y con resultados, cuya propuesta se haya enfocado a la implementación de sistemas IoT con Blockchain. Se descartaron los trabajos sin una metodología clara, sin resultados y que no se hayan relacionado con Blockchain.

Siguiendo la investigación de Rekha et al. [1], quienes han demostrado la adaptación de la Internet de las Cosas (IoT) de varios dispositivos en numerosas industrias, particularmente en la industria de la medicina, donde los volúmenes de datos transmitidos por estos dispositivos son grandes y además no hay garantía de que los datos y los dispositivos estén seguros. El objetivo principal ha sido extraer evidencia de dispositivos Raspberry Pi y registrarla en el libro de contabilidad digital. Esta investigación hace ver la importancia de la protección de los datos ya que si esto se logra vulnerar puede resultar en la muerte de varios pacientes donde los datos no deben ser vulnerados. Dentro de este contexto es importante resaltar la poca seguridad que tienen los dispositivos Raspberry PI.

Luego, se encuentra la investigación de Román et al. [2] quien ha propuesto un sistema de IoT que recopila, envía, almacena y publica todos los datos más relevantes utilizando dispositivos Raspberry PI como sensores inteligentes. La información se almacena en tecnologías distribuidas de BigchainDB e IOTA, que son bases de datos similares a una cadena

de bloques. Este sistema se puede convertir en una fuente de evidencias para diferentes tipos de empresas interesadas en proteger sus datos a través de tecnologías alternativas logrando obtener beneficios al mantener el nivel de confianza a lo largo de la cadena de valor.

Siguiendo con la investigación, se encuentra el artículo presentado por Junfithrana et al. [3] quienes han propuesto la idea de crear un sistema de donación de arroz basado en IoT y blockchain, en donde la existencia de arroz en orfanatos puede ser detectadas por dispositivos Raspberry pi con la condición de que estén conectados a sensores. Han logrado diseñar este sistema con una aplicación móvil, que ha sido diseñada para proveedores de servicios y arroz. Los donantes de arroz envían a través de transacciones financieras con proveedores de servicios y los proveedores de arroz envían los artículos a los orfanatos. En este artículo, la tecnología blockchain se ha implementado con el objetivo de que todos los aportes deben ser transparentes y reducir la manipulación de transacciones.

Por otro lado, se rescata el trabajo realizado por Kaushik et al. [4] en donde se ha diseñado un sistema de monitoreo con dispositivos IoT encargados de recopilar la temperatura y la humedad de contenedores de mercancías, además, realiza un seguimiento cuando la tapa del contenedor se abrió, en un determinado momento o no, y el tiempo que permaneció abierta. El sistema se ha implementado en una Raspberry Pi 4 que se encargó de todo el monitoreo y seguimiento de los demás sensores, la tecnología blockchain se implementó para registrar cuando la tapa del contenedor permanece abierta y, a través de transacciones, se puede ver quién y cuando hizo uso del contenedor.

En el trabajo presentado por Mohammed et al. [5] se ha detallado los componentes de la arquitectura y una solución adecuada que utiliza un enfoque experimental basado en el análisis de video, han implementado soluciones de IoT que utilizan tecnología blockchain de Hyperledger Sawtooth. El avance del Internet de las Cosas (IoT) ha impulsado un aumento de los dispositivos conectados, en donde se genera un flujo enorme de datos que requiere de recursos considerables de análisis y procesamiento. Procesar los datos más cerca de donde se generan suena bien, pero instalar toda la maquinaria necesaria en ese punto a menudo es complicado, por lo que el trabajo brinda la idea de usar tecnología blockchain que permita un procesamiento más cercano a los dispositivos IoT.

Los estudios realizados por Gangothri et al. [6] han propuesto una arquitectura de atención médica ambiental basada en sensores utilizando blockchain e IoT. La atención médica es un área con mucho movimiento que adopta la tecnología para dar tratamientos eficientes al paciente. Muchas tecnologías actuales como la Inteligencia Artificial (IA), IoT y blockchain brindan un amplio catálogo de aplicaciones que satisfacen las necesidades de atención médica.

Este trabajo ha propuesto una arquitectura que incluye Inteligencia Artificial en la plataforma de atención médica. Además, se ha utilizado blockchain como base principal del sistema que incluye un almacenamiento seguro de los datos entre diferentes partes de la red. Por último, se ha incluido un prototipo de hardware que se basa en un modelo de Raspberry Pi 4B en donde se conectan todos los sensores que capturan los datos vitales del cuerpo.

El trabajo propuesto por Gong-Guo et al. [7] ha presentado un marco de cadena de bloques para la calidad de datos IoT a través de la computación perimetral. Una casa inteligente presenta un desafío en el control y monitoreo de sus redes de sensores inalámbricos (WSN) y los dispositivos de IoT que lo contienen. Muchas de las arquitecturas de IoT manejan una arquitectura centralizada y son complejas con una seguridad baja en las comunicaciones. Debido a esto, existen problemas de integridad de los datos, estos problemas contienen datos faltantes, inserción de datos maliciosos y sobrecarga en la red de comunicaciones, además de una sobrecarga en sus nodos principales. Este trabajo brinda una nueva arquitectura que se basa en blockchain, introduciendo la capa de computación en el borde y se ha propuesto un nuevo algoritmo para mejorar la calidad de los datos y detectar datos maliciosos o falsos.

En el estudio presentado en Dharani et al. [8] se ha explorado que los dispositivos IoT tienen características de rendimiento y movilidad limitados en ciertas zonas, esto dificulta los procesos de seguridad que permiten el ingreso o autenticación. El estudio ha propuesto un sistema de autenticación de seguridad con el nombre IoT-Chain que se basa en atributos. El sistema tiene cuatro tipos de código: de cadena, de acceso, de dispositivo y código de política. Todos estos códigos se implementan y proporcionan la estrategia de control de acceso para el usuario administrador. Se combina el control de acceso y la tecnología blockchain que brinda una gestión dinámica de la autenticación de seguridad de IoT. Los resultados demuestran que esta combinación, puede mantener un alto rendimiento y alcanzar una validación efectiva en un sistema de control de autenticación.

En cualquier entorno corporativo, surgen amenazas inevitables. Por esta razón, el artículo propuesto por Mohammed et al. [9] ha sugerido la integración de blockchain con IoT como medida para mitigar las amenazas cibernéticas. La propuesta se fundamenta en un modelo eficiente y altamente seguro que busca abordar las limitaciones de los recursos al ser consumidos. Los dispositivos IoT generan una cantidad significativa de datos valiosos y confidenciales, los cuales se comparten con otras aplicaciones externas para ofrecer servicios útiles. Los sistemas tradicionales de control de acceso basados en IoT no involucran a todas las partes interesadas en el proceso de toma de decisiones. La integración de blockchain con IoT proporciona una capa adicional de seguridad al garantizar la inmutabilidad de los datos y la

transparencia en todas las transacciones. Este enfoque no solo fortalece la protección contra amenazas cibernéticas, sino que también mejora la confianza entre las partes involucradas.

Según Paret et al. [10] los dispositivos IoT presentan limitaciones en cuanto a rendimiento y movilidad, lo que complica la aplicación de métodos convencionales de autenticación de seguridad centralizada en el actual entorno de IoT. Con el objetivo de abordar estos desafíos, se ha planteado en este artículo la propuesta de un sistema de autenticación de seguridad denominado "iot-chain" para el Internet de las Cosas. Este sistema se fundamenta en la autenticación de seguridad basada en atributos, utilizando el marco de la cadena de bloques Hyperledger Fabric.

A continuación, en la **TABLA I** se resume la información extraída de cada una de las propuestas, con el objetivo de comprender mejor la idea central, sus conclusiones y los métodos que se han propuesto.

TABLA I
COMPARACIÓN DE INVESTIGACIÓN BIBLIOGRÁFICA

Tesis, Artículos Científicos, Investigaciones.	Idea Principal	Conclusiones	Método(s) implementado
Investigación forense de Raspberry Pi y preservación de pruebas mediante blockchain [1].	Extraer evidencia de Raspberry Pi y registrarlas en un libro de contabilidad digital.	Se ha comprobado el gran impacto de la blockchain con IoT.	Extraer evidencia de Raspberry Pi y registrarlas en un libro de contabilidad digital.
Tecnologías IoT blockchain para sensores inteligentes basados en Raspberry Pi [2]	Crear un sistema de IoT que recopila, envía y almacena datos.	La información se almacena en tecnologías distribuidas de BigchainDB	Base de datos BigchainDB e IOTA. Extracción, almacenamiento y envío de datos.
Sistema de donación de arroz en orfanato basado en Internet de las Cosas [3]	Crear un sistema de donación de arroz basado en IoT y Blockchain.	Se ha diseñado el sistema con una aplicación móvil.	Tecnología blockchain para las transacciones.
Sistema de monitoreo de inventario basado en blockchain e IoT [4]	Sistema de monitoreo con dispositivos IoT que se encarga de recopilar la temperatura y la humedad de contenedores de mercancías	La tecnología blockchain se ha implementado al momento de abrir o cerrar el contenedor, a través de transacciones se puede ver quién y cuando hizo uso del contenedor.	Raspberry Pi 4 que se encargó de todo el monitoreo y seguimiento de los sensores.

TABLA II
(CONTINUACIÓN) COMPARACIÓN DE INVESTIGACIÓN BIBLIOGRÁFICA

Tesis, Artículos Científicos, Investigaciones.	Idea Principal	Conclusiones	Método(s) implementado
Computación perimetral de IoT habilitada para blockchain [5]	Solución adecuada que utiliza un enfoque experimental basado en el análisis de video en el borde, implementa soluciones de IoT que utilizan tecnología blockchain de Hyperledger Sawtooth	La implementación de lockchain permite un procesamiento más cercano con IoT.	Análisis de video en el borde. Blockchain de Hyperledger Sawtooth.
Arquitectura de salud ambiental basada en sensores utilizando Blockchain e Internet de las cosas [6]	Arquitectura de atención médica ambiental basada en sensores utilizando blockchain e IoT.	Se ha logrado la integración de Inteligencia Artificial en plataformas de atención médica.	Se ha utilizado blockchain como base principal del sistema, incluye un almacenamiento seguro de los datos e intercambio entre diferentes partes de la red.
Sistema de autenticación de seguridad de IoT basado en blockchain [8]	Características de rendimiento y movilidad limitados en ciertas zonas de IoT.	El sistema tiene tres tipos de código de cadena, código de acceso, código de dispositivo y código de política	Control de acceso. Tecnología Blockchain.
Integración de Blockchain con IoT como medida para mitigar las amenazas cibernéticas [9]	Modelo eficiente y altamente seguro que aborda las limitaciones de los recursos al ser consumidos	La integración de blockchain con IoT proporciona una capa adicional de seguridad al garantizar la inmutabilidad de los datos	Integración de Blockchain. Control de acceso basados en IoT.
Marco de cadena de bloques para la calidad de los datos de IoT a través de la computación perimetral [10]	Nueva arquitectura que se basa en blockchain, introduce la capa de computación en el borde y se ha propuesto un nuevo algoritmo para mejorar la calidad de los datos	Se ha corregido en cierta medida la integridad de los datos, además de la detección temprana de datos maliciosos o falsos.	Computación perimetral. Arquitectura centralizada.
Sistema de autenticación de seguridad denominado "iot-chain" para el Internet de las Cosas [11]	IoT presenta limitaciones en cuanto a rendimiento y movilidad, lo que complica la aplicación de métodos centralizada en el actual entorno.	Se ha implementado el sistema “iot-chain” reduciendo la implicación de rendimiento y movilidad, aplicando la tecnología de cadenas de bloques.	Marco de la cadena de bloques. Hyperledger Fabric.

1.2. Objetivo general

- Diseñar e implementar un sistema de control de ingreso utilizando un backbone IoT y tecnología Blockchain.

1.3. Objetivos específicos

- Implementar un Backbone IoT para evaluar la funcionalidad e integración de dispositivos de tipo Raspberry PI.
- Evaluar, seleccionar y configurar los tags de identificación/detección y dispositivos lectores basados en tecnologías RFID.
- Implementar la Tecnología Blockchain mediante la evaluación y selección de la plataforma adecuada para la gestión de registros de acceso.
- Configurar y desplegar la red Blockchain para asegurar la inmutabilidad de los datos.
- Integrar la plataforma Blockchain con el sistema IoT para garantizar la trazabilidad de la información de registros de acceso.

1.4. Problemática

La situación actual en la Universidad de las Fuerzas Armadas ESPE, sede Santo Domingo, plantea un panorama que presenta problemas al momento de controlar el acceso de estudiantes, personal civil, militar y vehículos. El proceso manual de control de ingreso implica una capacidad limitada para manejar el flujo constante de personas y vehículos, especialmente en franjas de alta demanda. La ausencia de tecnologías específicas para el control de ingreso implica mayor probabilidad de errores humanos y limita la capacidad de realizar un seguimiento a los datos históricos.

Uno de los principales problemas identificados radica en la dependencia de recursos humanos para llevar a cabo el registro de vehículos y la identificación de personas. Este enfoque manual se traduce en tiempos de espera prolongados, congestiones en la entrada y salida de la universidad, y la posibilidad de errores humanos que comprometen la integridad de los datos registrados. Además, la falta de automatización dificulta la gestión eficiente de la información y la generación de reportes en tiempo real.

Otro aspecto crítico es la seguridad. El método manual actual no proporciona un nivel óptimo de control y verificación de identidad, lo que podría resultar en accesos no autorizados o la entrada de personas no registradas en el sistema. Esta falta de rigurosidad en la identificación representa un riesgo potencial para la seguridad interna de la institución.

Adicionalmente, la universidad cuenta con un componente militar, lo que agrega complejidad al proceso de control de ingreso. La necesidad de coordinar y gestionar eficientemente la entrada y salida de personal militar complica aún más el sistema actual, destacando la urgencia de una solución más avanzada y tecnológica.

En este contexto, la implementación de un sistema basado en IoT y Blockchain se presenta como una alternativa prometedora para superar estas limitaciones. La adopción de estas tecnologías innovadoras permitirá no solo agilizar y mejorar la precisión del control de ingreso, sino también reforzar la seguridad y facilitar la gestión integral de la información asociada a este proceso.

1.5. Justificación

Esta propuesta se justifica en virtud de la necesidad de optimizar y modernizar el actual sistema manual de control de acceso en la Universidad. Nuestro trabajo se sustenta en diversos aspectos que abordan tanto las limitaciones actuales como las oportunidades que las tecnologías IoT y Blockchain ofrecen para mejorar significativamente este proceso crítico.

- ***Optimización de Eficiencia:*** La implementación de un sistema basado en IoT permitirá la automatización del proceso de registro manual, reduciendo los tiempos de espera, eliminando cuellos de botella y agilizando la circulación dentro de la universidad.
- ***Mejora en Seguridad:*** La adopción de la tecnología basada en blockchain proporcionará un nivel adicional de seguridad al garantizar la integridad y la confidencialidad de los datos.
- ***Gestión Integral de la Información:*** La incorporación de tecnologías IoT y blockchain facilitará la recopilación, almacenamiento y análisis de datos, permitiendo una gestión integral de la información asociada al control de ingreso.
- ***Incorporación de Innovación Tecnológica:*** En un entorno académico y tecnológico en constante evolución, la implementación de un sistema basado en IoT y blockchain posicionará a la universidad a la vanguardia de la innovación tecnológica. Esto contribuirá a mejorar la imagen institucional y fortalecerá la posición competitiva de la universidad en el ámbito educativo.

2. Marco teórico

2.1. Blockchain

Según la publicación realizada por Guaña-Moya et al. [11] se define a la blockchain como una forma de tecnología de contabilidad distribuida (DLT) que se encarga de registrar bloques de información en la red de la cadena siendo está respaldada por nodos que la conforman.

Para entender mejor este significado, pongamos como ejemplo, en un salón de clases el docente crea una actividad la cual consiste en decir una palabra cualquiera. El estudiante inicial menciona su palabra y finaliza su turno, después, el segundo estudiante debe mencionar tanto la palabra elegida por su compañero como uno nuevo, y este proceso continúa secuencialmente.

2.2. Bloques

Guaña-Moya et al. [11] afirma que, los bloques se generan a través de un software de código abierto donde se guarda la información sobre cuándo y en qué orden se ha llevado a cabo la transacción. Este segmento almacena de manera cronológica los datos generados por las transacciones dentro de la cadena, razón por la cual se le denomina cadena de bloques o blockchain. En el ejemplo anterior, las palabras de los estudiantes serían los bloques de la información, mientras que las palabras unidas de manera cronológica se la pueden ver como la cadena.

2.3. Hashing

La tecnología de blockchain [11] utiliza un método para unir los bloques llamado hashing, que se encarga del proceso de encriptar toda la información haciendo uso de funciones matemáticas. En otras palabras, se trata del proceso que confiere a un bloque su propia identidad única y no duplicable, que se asemeja a una huella digital conocida como hash. Cuando se incorpora un bloque adicional a la cadena, comienza un procedimiento en el que se incluye el hash del bloque más reciente de la cadena. De esta forma, el hash se registra tanto en el bloque ya existente como en el nuevo bloque que se añade. Si se produce alguna alteración en los datos almacenados en un bloque, el hash de ese bloque cambiará automáticamente.

2.4. Mecanismo de Consenso

Se le conoce como el método de validación el cual, se refiere al proceso en el cual los nodos verifican que la información administrada es válida [11].

2.5. *Tipos de Blockchain*

Según la investigación realizada por [12] existen tres tipos diferentes de blockchain. Cuando una blockchain permite ver toda la información dentro de la cadena se trata de una blockchain pública. Si solo se puede visualizar por ciertas personas se trata de una blockchain tipo privada. Por otra parte, se encuentra la blockchain permissioned la cual permite que solo un grupo selecto de usuarios genere información y la valide. Por último, las blockchain permissionless, permiten que cualquier persona contribuya y valide la información.

2.6. Internet de las cosas (IoT)

Según el artículo publicado por Moreno [12] afirma que la internet de las cosas (IoT) representa la próxima evaluación de la internet ya que tiene un gran avance en su capacidad de recopilar, analizar y distribuir datos que se pueden convertir en información y conocimiento.

Luego [12] asegura que IoT se refiere a la consolidación a través de una red de redes que alberga una amplia variedad de dispositivos, donde la red tiene la capacidad de conectar múltiples dispositivos. Una definición más estructurada es la que dice [11] donde propone su idea de que el término IoT hace referencia a los sistemas de dispositivos físicos con mínima intervención humana, intercambian datos mediante conexiones inalámbricas.

2.7. Radiofrecuencia de Identificación (RFID)

Según el informe proporcionado por el portal de Amsler et al. [13] sostiene que la tecnología de Identificación por Radiofrecuencia (RFID) permite la transmisión de datos a través de ondas de radiofrecuencia. Esta transmisión se lleva a cabo exclusivamente mediante una etiqueta RFID, la cual posee la capacidad de almacenar, transmitir e identificar los datos asociados al objeto en cuestión.

2.8. Raspberry pi

La información recopilada por el sitio web MCI (https://raspberrypi.cl/que-es-raspberry/) la define como una serie de minicomputadoras de placa única desarrollada por Raspberry Pi Fundation. Entre sus componentes, cuenta con un procesador, memoria RAM, puertos de entrada y salida, su nombre nació por el género de Rubus, un tipo de fruta.

2.9. Tipo de Raspberry pi

En este apartado se explica el modelo de la raspberry pi utilizada en el proyecto y las características que contiene.

2.10. Raspberry Pi 4 modelo B

1. Mejoras en el puerto HDMI.
2. Resolución de 4K.
3. USB 3.0
4. Nuevo procesador tres veces mejor al anterior.
5. Memoria RAM de 4GB.
6. Almacenamiento Micro SD 64GB.

2.11. Ethereum

Ethereum es una plataforma descentralizada de código abierto basada en blockchain que permite la ejecución de contratos inteligentes y aplicaciones descentralizadas. Ethereum se destaca por su capacidad de ofrecer una capa de abstracción adicional sobre la tecnología blockchain, permitiendo a los desarrolladores crear y desplegar contratos inteligentes de manera eficiente [14].

2.12. Contratos Inteligentes

En el sitio web Chainlink [15] se refieren a los contratos inteligentes como programas informáticos ejecutables que utilizan protocolos de blockchain para facilitar, verificar o hacer cumplir la ejecución de acuerdos contractuales sin la necesidad de intermediarios tradicionales. Cada contrato está diseñado para asegurar y automatizar cláusulas contractuales de manera inmutable y descentralizada. La tecnología blockchain permite que los contratos inteligentes trabajen de manera confiable, eliminando dependencia de terceros y mitigando conflictos.

2.13. Solidity

Según la información proporcionada por Simplilearn [16], solidity se refiere a un lenguaje de programación diseñado para crear contratos inteligentes en plataformas basadas en blockchain, siendo Ethereum una de las más utilizadas. Es desarrollado para ser ejecutado en la máquina virtual de Ethereum, y permite a los desarrolladores codificar lógica empresarial y reglas contractuales de manera segura y verificable.

2.14. Truffle

Truffle se desempeña como un conjunto de herramientas de desarrollo integral que se encarga de facilitar y agilizar el proceso de creación, prueba y despliegue de contratos inteligentes en plataformas basadas en blockchain, con un enfoque primario en Ethereum. Este entorno brinda utilidades que permiten a los desarrolladores gestionar eficientemente todo el ciclo de vida del desarrollo de contratos inteligentes [17].

2.15. Metamask

Metamask es una extensión de navegador y cartera digital que permite a los desarrolladores y usuarios interactuar de forma segura con aplicaciones descentralizadas basadas en blockchain en específico de Ethereum. Metamask está desarrollada con el objetivo de simplificar la experiencia de usuario en el entorno descentralizado, facilita la gestión de claves privadas y firma las transacciones sin necesidad de ejecutar un nodo completo de Ethereum [18].

2.16. Ganache

Weston [19] se refiere a Ganache como una herramienta de desarrollo que simplifica la creación, prueba y despliegue de aplicaciones descentralizadas y contratos inteligentes basados en Ethereum. Está desarrollado por Truffle, y proporciona un entorno de desarrollo personal local que simula una cadena de bloques Ethereum, permitiendo a los desarrolladores realizar pruebas exhaustivas de contratos inteligentes en entornos controlados sin necesidad de interactuar con la red principal.

3. Metodología

3.1. Árbol de problemas para determinar causas y efectos

Se ha propuesto un análisis a través de un Árbol de Problemas para ilustrar (**Fig. 1**) las causas y efectos sobre la implementación de nuestra propuesta relacionada con el sistema de control de ingreso de estudiantes integrando las tecnologías IoT y Blockchain:

Problema Central: Implementación de un sistema de control de acceso de estudiantes usando IoT y blockchain.

A. Causas:

i. Falta de comprensión sobre la tecnología:

- Limitado conocimiento sobre IoT y blockchain entre el personal encargado de la implementación y el personal de seguridad física.
- Dificultades para entender cómo integrar ambas tecnologías de manera efectiva.

ii. Desafíos de interoperabilidad:

- Incompatibilidad entre diferentes dispositivos IoT utilizados en el sistema de control de acceso.
- Dificultades para integrar la tecnología blockchain con sistemas existentes de gestión de ingreso de estudiantes.

iii. Seguridad y privacidad:

- Vulnerabilidades en la seguridad de los dispositivos IoT que podrían ser explotadas por terceros.
- Preocupaciones sobre la privacidad de los datos almacenados en la cadena de bloques y la posibilidad de violaciones de datos.

B. Efectos:

i. Retrasos en la implementación:

- Dificultades técnicas que resultan en un retraso en el lanzamiento del sistema de control de acceso.
- La necesidad de resolver problemas de interoperabilidad y seguridad prolonga el tiempo de implementación.

ii. Frustración del usuario:

- Experiencias inconsistentes y problemas técnicos pueden llevar a la insatisfacción de los usuarios finales.
- Los estudiantes y el personal pueden experimentar dificultades para acceder a las instalaciones debido a fallos en el sistema.

iii. Riesgos de seguridad:

- Brechas de seguridad podrían comprometer la integridad de los datos de los estudiantes y del personal.
- La exposición a ciberataques podría resultar en la pérdida o el robo de información confidencial.

iv. Pérdida de confianza:

- Fallos recurrentes en el sistema podrían afectar la confianza en la tecnología y en la universidad que la implementa.
- La percepción negativa sobre la seguridad y la eficacia del sistema podría dañar la reputación de la universidad.

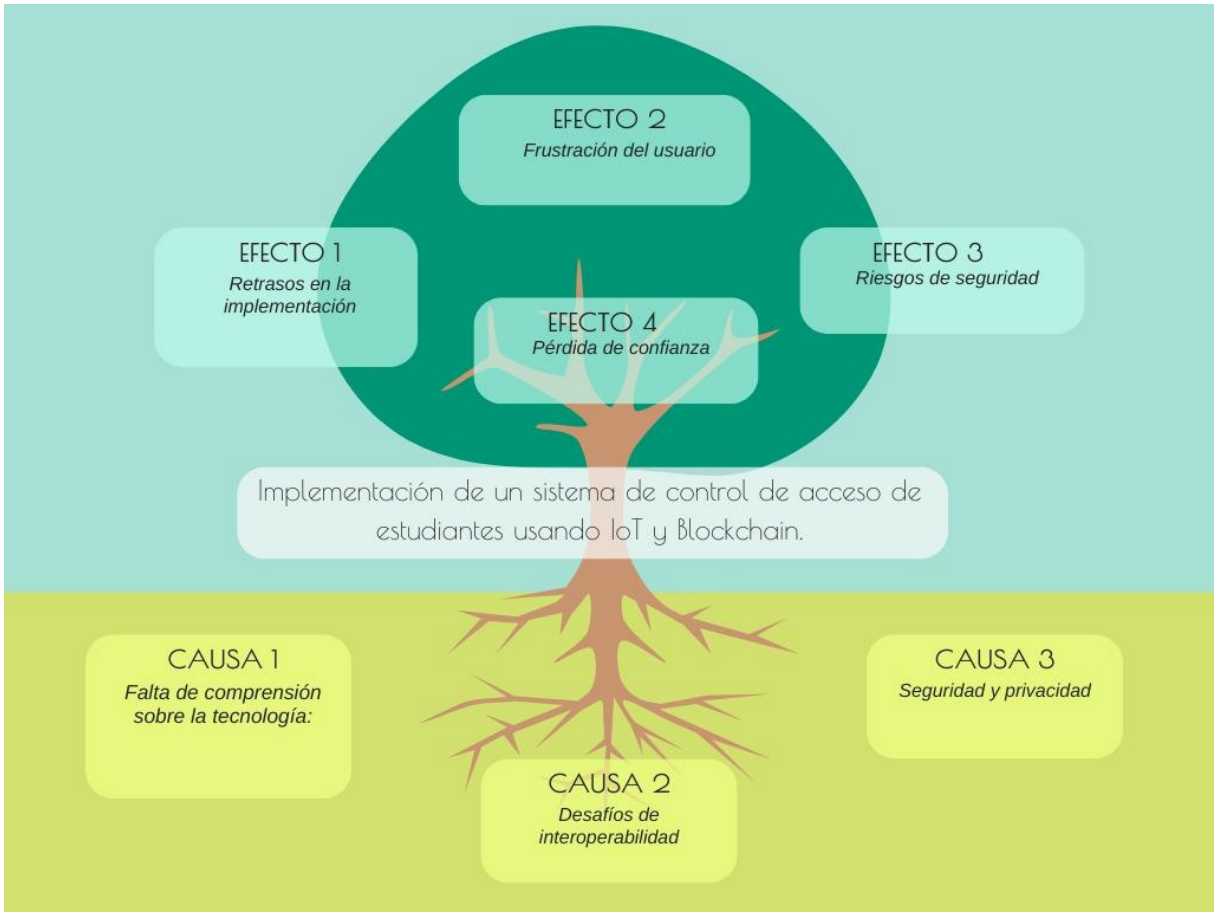


Fig. 1 Árbol de problemas para identificar las causas y los efectos.

3.2. Modelado de Amenazas en Tecnologías IoT

El modelado de amenazas funciona identificando los tipos de amenazas que causan o podrían causar daño a una aplicación o sistema informático. Hemos adoptado la perspectiva de un pirata informático malintencionado para ver cuánto daño podría hacer sobre nuestra propuesta.

Con este modelado de amenazas (**Fig. 2**), hemos realizado un análisis de la arquitectura IoT propuesta, el contexto universitario de su implementación y otra información complementaria (por ejemplo, especificaciones técnicas de los dispositivos para identificar vulnerabilidades). Este proceso nos ayudó con la comprensión más profunda y el descubrimiento de amenazas de todo el sistema.

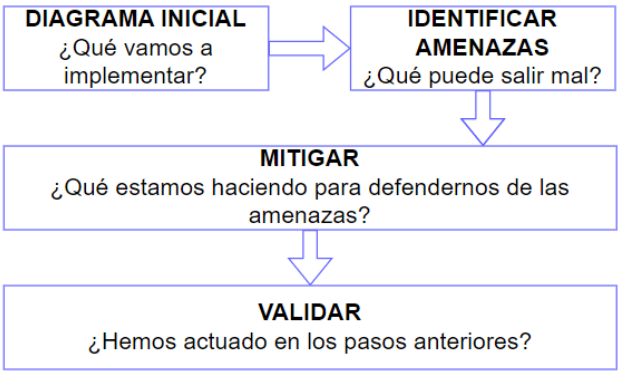


Fig. 2 Modelado de amenazas aplicado a la propuesta de integración Iot+Blockchain

Diagrama Inicial: aquí se describe lo que vamos a implementar, en resumen, se integrarán tecnologías IoT y blockchain para validar un sistema de control de ingreso de estudiantes a la Universidad de las Fuerzas Armadas ESPE sede Santo Domingo de los Tsáchilas. Se usará un control mediante tags y un lector RFID conectados sobre una infraestructura de red en malla.

Identificar Amenazas: aquí se describe lo que podría salir mal, en el contexto universitario, los datos transmitidos entre los dispositivos IoT y el servidor central podrían ser interceptados por atacantes, lo que podría comprometer la confidencialidad de la información (filtrado de datos de identificación de los estudiantes). Dentro de un sistema de control de acceso tradicional un atacante podría inyectar comandos maliciosos en los dispositivos IoT o en el servidor central, lo que podría conducir a comportamientos no deseados o incluso a la toma de control completo del sistema.

La infraestructura física propuesta puede ser comprometida por cualquier tipo de amenaza, por ejemplo, los módulos Raspberry Pi al igual que cualquier otro dispositivo informático, pueden verse afectados por vulnerabilidades de software, como fallas en el sistema operativo o aplicaciones maliciosas que pueden comprometer la seguridad del sistema. Además, los atacantes podrían intentar acceder a la Raspberry Pi mediante ataques de fuerza bruta, intentando adivinar contraseñas débiles o predecibles.

El módulo RFID PN532 podría ser susceptible a la manipulación de su firmware, permitiendo que un atacante pueda modificar su comportamiento para realizar acciones maliciosas, como la clonación de tarjetas de acceso. Las tarjetas y los tags RFID podrían ser clonadas por un atacante, permitiendo el acceso no autorizado al sistema utilizando tarjetas falsificadas.

Debido a la naturaleza inalámbrica de una red en malla, existe el riesgo de que las comunicaciones entre los dispositivos puedan ser interceptadas por un atacante, lo que podría exponer datos sensibles o permitir ataques de tipo Man in the Middle (MITM), en donde un atacante intercepta y potencialmente altera la comunicación entre dos partes sin su conocimiento. En este tipo de ataque, el atacante se posiciona entre el remitente y el destinatario, interceptando los mensajes y, en algunos casos, modificándose antes de enviarlos al destinatario original.

Dentro del sistema operativo, los servicios y demonios que se ejecutan en el sistema operativo Ubuntu Server, como el servicio SSH, Apache, o servicios de base de datos, podrían contener vulnerabilidades que podrían ser explotadas por un atacante para obtener acceso no autorizado al sistema o comprometer la integridad de los datos.

Mitigar: Nuestra propuesta debe asegurar 2 pilares de la triada de seguridad informática (CIA), que corresponde a la Confidencialidad e Integridad de los datos que circulen por la infraestructura que se implemente.

Validar: Nuestra infraestructura funciona correctamente desde los dispositivos conectados entre sí con la red en malla (mesh). Pasando por la lectura y envío del ID de un tag o tarjeta personal a los nodos, encriptando los datos en cadenas de bloques (blockchain). Se validará el control de acceso a través de una aplicación web que permite el registro de estudiantes mediante el escaneo de su tag/tarjeta RFID. Permite buscar estudiantes a través de un datatable y mediante el ingreso de cualquier dato. En la misma tabla existe un botón de editar donde se da clic y se cargan los datos en el formulario para posteriormente actualizarlos.

4. Diseño

Nuestra propuesta se basa en un diseño integral que comprendió diversas etapas, como se muestra en la (**Fig. 3**). Se inició con la instalación del sistema operativo en dispositivos embebidos tipo Raspberry Pi Modelo 4, seguido de una minuciosa evaluación de las características de almacenamiento, procesamiento y memoria. Luego, se implementó una red en malla (mesh) y se realizaron pruebas exhaustivas para verificar la conectividad entre los nodos de la mesh. Posteriormente, se integró el módulo PN532 RFID y se ejecutaron pruebas para garantizar la lectura y el envío adecuado del ID de los tags/tarjetas al lector RFID. Finalmente, se agregó un nivel adicional de seguridad al sistema mediante la implementación de encriptación utilizando blockchain. Este diseño abordó de manera integral los aspectos críticos de instalación, implementación, evaluación, y seguridad de todo el sistema propuesto.

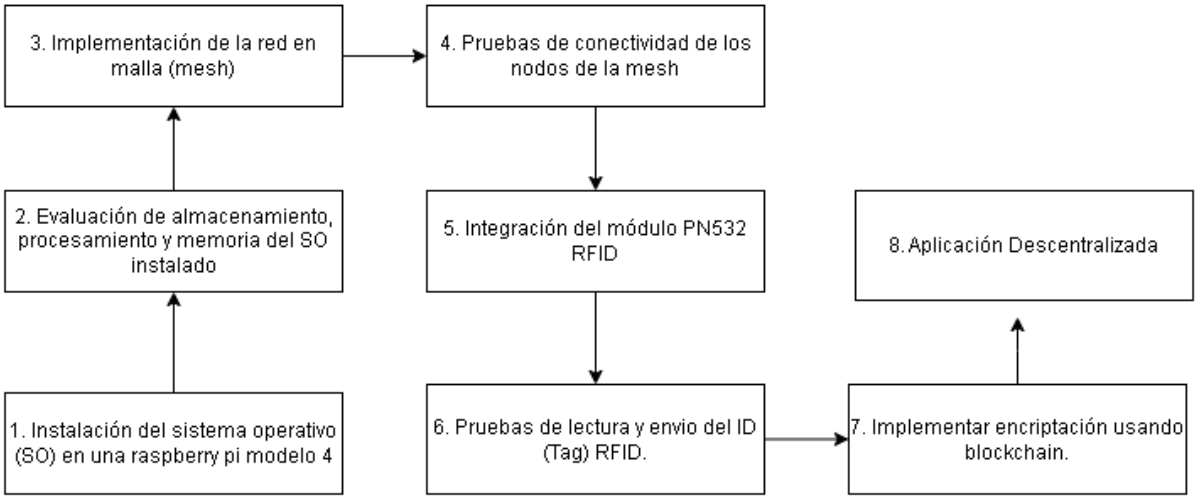


Fig. 3 Diseño de la propuesta para integrar las tecnologías IoT + blockchain.

A continuación, se explica detalladamente cada una de las etapas mencionadas en la (**Fig. 3**).

4.1. Instalación del sistema operativo (SO)

Se instaló y evaluó el sistema operativo en cada uno de los módulos Raspberry Pi con el objetivo de comprobar cuál es mejor en temas de rendimiento, almacenamiento y memoria. Para ello, se instalaron desde las fuentes oficiales las imágenes de los sistemas Ubuntu 20.04 Desktop (**Fig. 4**) y Ubuntu 20.04 Server (**Fig. 5**).

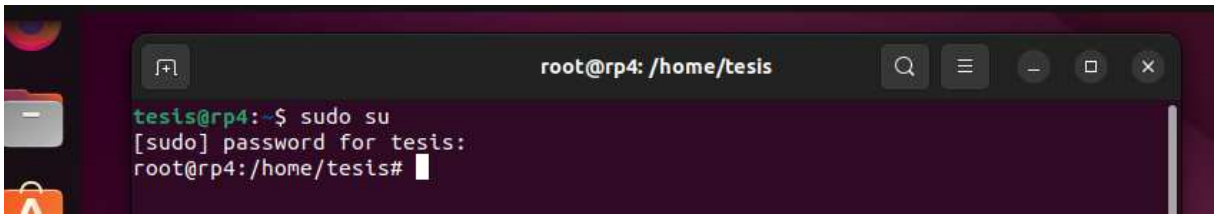


Fig. 4 Raspberry Pi 4 corriendo con SO Ubuntu 20.04 Desktop

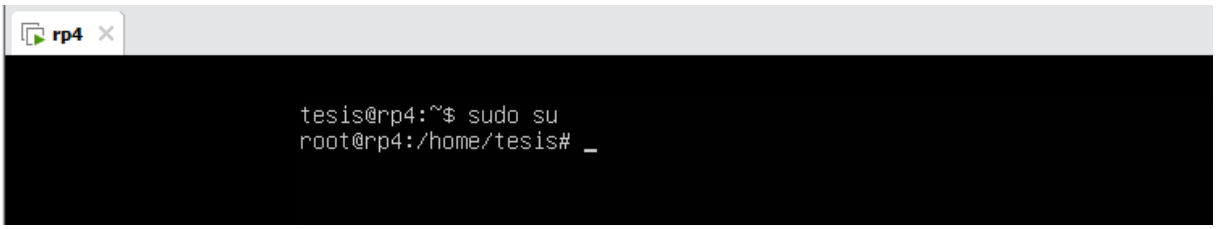


Fig. 5 Raspberry Pi 4 corriendo con SO Ubuntu 20.04 Server

4.2. Evaluación de almacenamiento, procesamiento y memoria de los SO instalados.

A continuación, se presentan algunos comandos ejecutados para verificar el rendimiento entre la imagen Ubuntu 20.04 Desktop y Ubuntu 20.04 Server.

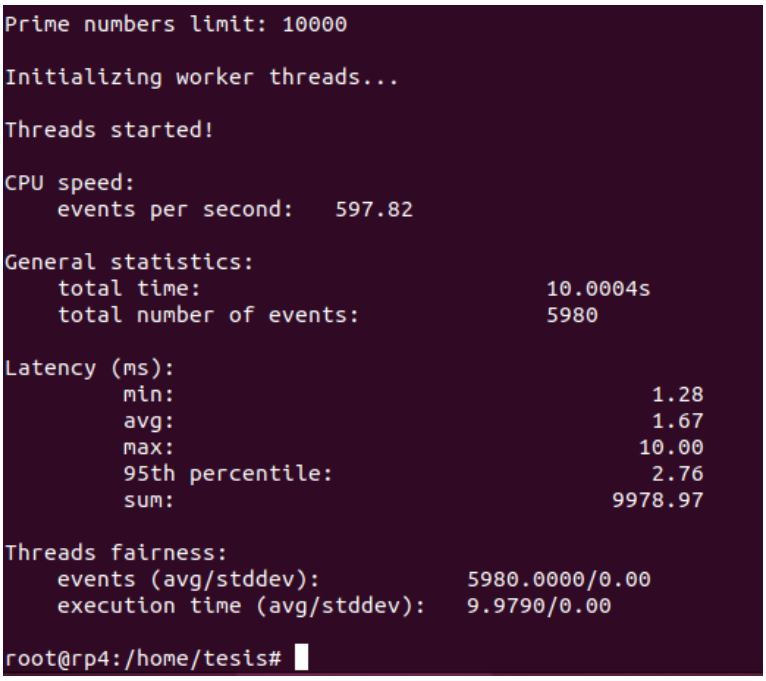


Fig. 6 Resumen de los parámetros de rendimiento del procesador en donde se instaló Ubuntu 20.04 Desktop

A continuación, se desglosa la información plasmada en la (**Fig. 6**).

- *CPU Speed:* Muestra la velocidad de la CPU en términos de eventos por segundo. En este caso, se realizaron 631.61 eventos por segundo.
- *General Staticts:* Indica el número total de eventos realizados durante la prueba, que fue de 6319 eventos.
- *Latency:* Proporciona estadísticas de latencia en milisegundos. Incluye valores mínimos (min), promedio (avg), máximo (max), percentil 95 (95th percentile), y la suma total (sum) de la latencia.
- *Threads fairness:* Muestra la equidad entre los hilos en términos de eventos y tiempo de ejecución. En este caso, el promedio de eventos por hilo fue 6319, y el tiempo de ejecución promedio por hilo fue 9,9841 segundos.

También, se midió el rendimiento en términos de memoria.

Fig. 7 Rendimiento de memoria en Ubuntu 20.04 Desktop

A continuación, se detalla la información plasmada en la (**Fig. 7**).

Memoria Principal (RAM)

- La cantidad total de memoria RAM (MB) en el sistema fué de 3870 MB.
- La cantidad de memoria RAM que se usó fué de 939 MB.
- La cantidad de memoria RAM que estuvo disponible para ser utilizada fué de 1437 MB.
- La cantidad de memoria que se compartió entre varios procesos fué de 29 MB.
- La cantidad de memoria utilizada como caché por el sistema operativo para acelerar el acceso a los datos almacenados en el disco fué de 1493 MB.
- La cantidad de memoria RAM que estaba disponible para procesos nuevos y se podía incluir parte de la caché fue de 2669 MB.

Área de Intercambio (Swap):

- La cantidad total de espacio de intercambio (swap) del sistema actual fue de 2139 MB.
- La cantidad de espacio de intercambio actualmente en uso. En este caso, fue de 0 MB, lo que significa que no se estaba utilizando.
- La cantidad de espacio de intercambio que estuvo disponible para ser utilizado fue de 2139 MB.

Estos fueron algunos de los comandos utilizados para ver el rendimiento de Ubuntu Desktop 20.04.

4.2.1 Instalación Ubuntu 20.04 server.

A continuación, se resume el proceso de ejecución de comandos para revisar el rendimiento en el CPU, el comando ***sysbech cpu --time=10 run*** ejecutó una prueba de rendimiento en la CPU durante 10 segundos y proporcionó información del procesamiento, como se observa en la (**Fig. 8**).

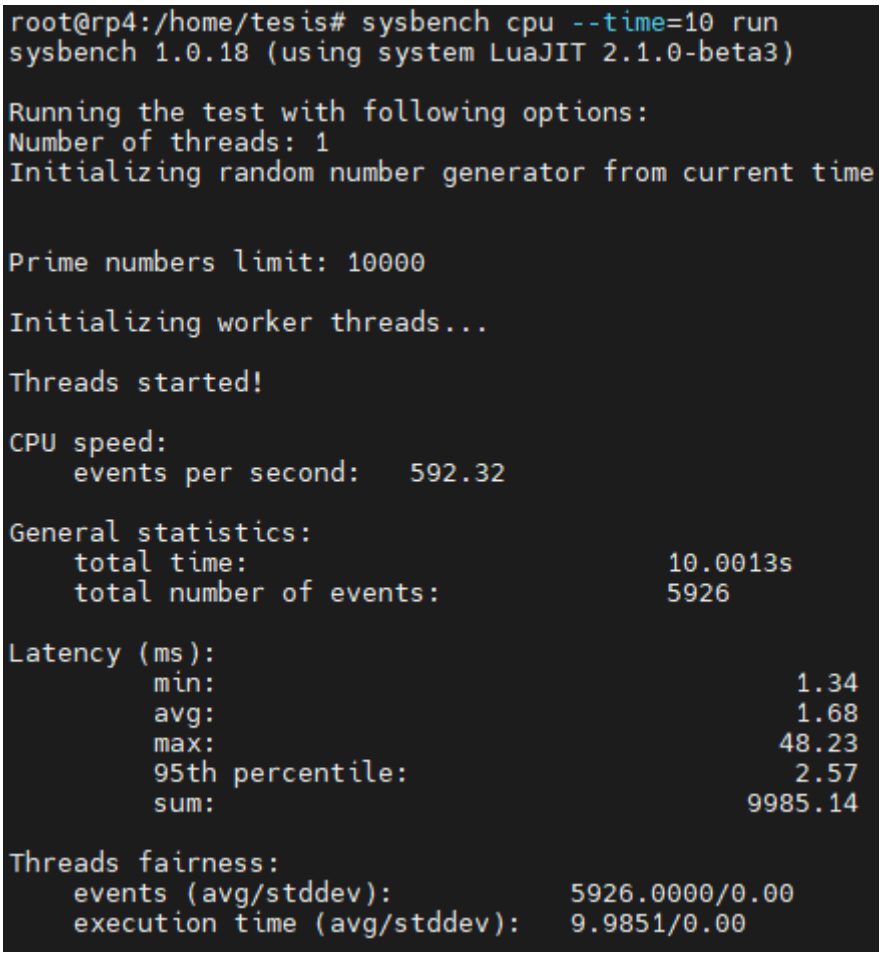


Fig. 8 Procesamiento de Ubuntu 20.04 Server

A continuación, se desglosa la información plasmada en la (**Fig. 8**).

- Muestra la velocidad de la CPU en términos de eventos por segundo. En este caso, se realizaron 592.32 eventos por segundo.
- Indica el número total de eventos realizados durante la prueba, que fue de 5926 eventos.
- Proporciona estadísticas de latencia en milisegundos. Incluye valores mínimos (min), promedio (avg), máximo (max), percentil 95 (95th percentile), y la suma total (sum) de la latencia dando un total de 9985.14 ms de latencia.
- Muestra la equidad entre los hilos en términos de eventos y tiempo de ejecución. En este caso, el promedio de eventos por hilo fue 5936, y el tiempo de ejecución promedio por hilo fue 9.9851 segundos.

A continuación, en la (**Fig. 9**), se hace un análisis a la memoria del sistema instalado

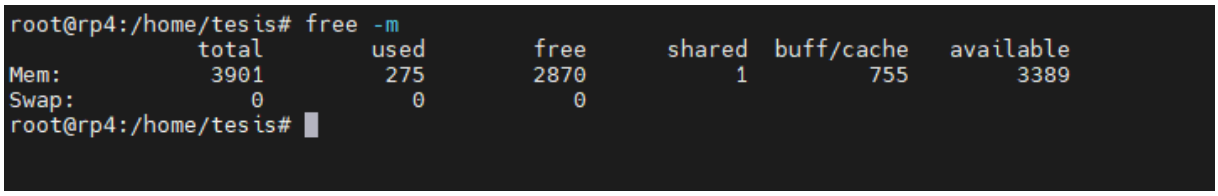


Fig. 9 Rendimiento en memoria de Ubuntu 20.04 Server

Memoria Principal (RAM)

- La cantidad total de memoria RAM en megabytes (MB) fue de 3901 MB.
- La cantidad de memoria RAM actualmente en uso fue de 275 MB.
- La cantidad de memoria RAM que estuvo disponible para ser utilizada fue de 2870 MB.
- La cantidad de memoria que se compartió entre varios procesos fue de 1 MB.
- La cantidad de memoria utilizada como caché por el sistema operativo para acelerar el acceso a los datos almacenados en el disco fue de 755 MB.
- La cantidad de memoria RAM que estuvo disponible para procesos nuevos y pudo incluir parte de la caché fue de 3389 MB.

Área de Intercambio (Swap):

- La cantidad total de espacio de intercambio (swap) en megabytes (MB) es este caso fue de 0 MB.
- La cantidad de espacio de intercambio actualmente en uso. En este caso, fue de 0 MB, lo que significa que no se utilizó.
- La cantidad de espacio de intercambio que estuvo disponible para ser utilizado fue de 0 MB

A continuación, en la (**TABLA II**) se muestra una comparativa entre las dos imágenes ISO (Ubuntu 20.04 Desktop y Ubuntu 20.04 Server) en temas de rendimiento de la CPU.

TABLA III

COMPARACIÓN ENTRE UBUNTU 20.04 DESKTOP Y UBUNTU 20.04 SERVER

Característica	**Ubuntu 20.04 Desktop**	**Ubuntu 20.04 Server**
CPU Speed (CPU velocidad)	597.82 eventos por segundos	592.32 eventos por segundo
General Staticts (Estadísticas generales)	5980 eventos	5926 eventos
Latency (Latencia)	9978.97 de latencia	9985.14 de latencia.
Threads fairness (Equidad de hilos)	5980 promedio de eventos por hilos	5936 promedio de eventos por hilos

En base al cuadro comparativo, se muestra de forma general, que Ubuntu 20.04 Server es mejor en temas de rendimiento de la CPU, aunque no presenta mucha diferencia comparada

con Ubuntu 20.04 Desktop, esta diferencia se hará notar con la ejecución de programas. Mientras que la instalación del SO Ubuntu 20.04 Desktop en una Raspberry pi, muestra que su CPU consume un total de 597.82 eventos por segundos, es decir, que se observará un poco de lentitud y esto se pudo comprobar con las estadísticas generales, llegando a 5980 eventos por segundo. Por lo tanto, en este proyecto, se implementará la imagen Ubuntu 20.04 Server.

A continuación, en la (**TABLA III**) se resume la comparación de los atributos de rendimiento de la memoria RAM entre Ubuntu 20.04 Desktop y Ubuntu 20.04 Server.

TABLA IV

COMPARATIVA MEMORIA RAM UBUNTU 20.04 DESKTOP Y UBUNTU 20.04 SERVER.

Característica	**Ubuntu 20.04 Desktop**	**Ubuntu 20.04 Server**
Cantidad de memoria utilizada	3870 MB	3901 MB
Memoria ocupada actualmente	939 MB	275 MB
Memoria Disponible	1437 MB	2870 MB
Memoria Compartida	29 MB	1 MB
Memoria en caché	1493 MB	755 MB
RAM disponible para procesos nuevos	2669 MB	3389 MB

4.3. Implementación de la red en malla (mesh)

En primera instancia, se creó un script con la extensión ".sh" que indicó que va a contener comandos de bash. Dentro del archivo se colocaron los comandos correspondientes a la configuración de la red mesh, como se observa en la (**Fig. 10**).

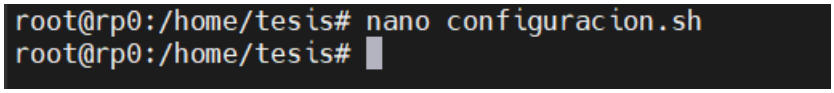


Fig. 10 Archivo de configuración red mesh.

Toda la configuración de la red mesh se enfocó en la interfaz wireless ("wlan0"), esto se observa en la (**Fig. 11**) en donde se detallan las configuraciones realizadas:

Primero, se apagó la interfaz de red inalámbrica wlan0. Luego, se configuró esta interfaz en modo ad-hoc, lo que significa que la red se creará sin un punto de acceso centralizado. Se estableció el canal de comunicación de la red ad-hoc en el canal 6 y se le asignó el nombre *'tesisBlock'* como SSID. Además, se configuró una clave de seguridad para la red ad-hoc, en este caso, la clave proporcionada fue '2350439861'.

A continuación, se asignó la dirección IP 192.168.22.2 a la interfaz wlan0, con una máscara de subred de 255.255.255.0, y luego se activó la interfaz wlan0.

Después, se añadió una ruta predeterminada a través de la puerta de enlace 192.168.22.1 para establecer cómo se debía enrutar el tráfico que no estaba destinado a la red local. Luego, se cargó el módulo del kernel para el protocolo BATMAN-adv, que es un protocolo de enrutamiento de red de mesh. Se añadió la interfaz wlan0 a BATMAN-Adv para que el protocolo pueda utilizarla para el enrutamiento de red. Finalmente, se configuró el modo de gateway de BATMAN-Adv en "server", lo que indicó que este nodo podía servir como puerta de enlace para otros nodos en la red de mesh BATMAN-Adv.

```
#!/bin/bash
ifconfig wlan0 down
iwconfig wlan0 mode ad-hoc
iwconfig wlan0 channel 6
iwconfig wlan0 essid 'tesisBlock'
iwconfig wlan0 key 2350439861
ifconfig wlan0 192.168.22.2 netmask 255.255.255.0 up
route add default gw 192.168.22.1
ifconfig wlan0 up
modprobe batman-adv
batctl if add wlan0
ifconfig wlan0 up
batctl gw_mode server
```

Fig. 11 Configuración red mesh

Se observa en la (**Fig. 12**) que la dirección IP de la interfaz wlan0 se encontraba habilitada y tenía la dirección 192.168.22.2, esta fue la IP del nodo principal de la red mesh.

```
root@rp0:/home/tesis# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.106  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::dea6:32ff:fea7:5a23  prefixlen 64  scopeid 0x20<link>
        ether dc:a6:32:a7:5a:23  txqueuelen 1000  (Ethernet)
        RX packets 14698  bytes 949980 (949.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 598  bytes 67450 (67.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 185  bytes 16712 (16.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 185  bytes 16712 (16.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.22.2  netmask 255.255.255.0  broadcast 192.168.22.255
        inet6 fe80::dea6:32ff:fea7:5a24  prefixlen 64  scopeid 0x20<link>
        ether dc:a6:32:a7:5a:24  txqueuelen 1000  (Ethernet)
        RX packets 7478  bytes 480285 (480.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4096  bytes 415352 (415.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@rp0:/home/tesis#
```

Fig. 12 Interfaz principal de la red mesh

Mediante el comando *iwconfig* se obtuvo información detallada sobre las interfaces inalámbricas, se observa en la (**Fig. 13**) que se encontraban los parámetros que contiene la interfaz wlan0, la cual fue la encargada de la red mesh.

```
root@rp0:/home/tesis# iwconfig
wlan0     IEEE 802.11  ESSID:"tesisBlock"
          Mode:Ad-Hoc  Frequency:2.437 GHz  Cell: 8E:55:7A:AA:15:57
          Tx-Power=31 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:2350-4398-61
          Power Management:on
```

Fig. 13 Interfaz WLAN 0

En la (**Fig. 14**) se observa el ingreso del comando *batctl o*, este comando se utilizó para mostrar información sobre los nodos vecinos en una red de mesh. Se visualizó la lista de los nodos que estaban directamente conectados al nodo local de la red mesh, se aprecia todos los saltos de la red y el último tiempo de recepción.

```
root@rp0:/home/tesis# batctl o
[B.A.T.M.A.N. adv 2019.4, MainIF/MAC: wlan0/dc:a6:32:a7:5a:24 (bat0/3e:5e:be:77:39:b5 BATMAN_IV)]
   Originator        last-seen (#/255) Nexthop           [outgoingIF]
   dc:a6:32:a7:59:d3    0.136s   (198) dc:a6:32:a7:5a:1e [     wlan0]
 * dc:a6:32:a7:59:d3    0.136s   (255) dc:a6:32:a7:59:d3 [     wlan0]
   dc:a6:32:a7:5a:1e    0.836s   (191) dc:a6:32:a7:59:d3 [     wlan0]
 * dc:a6:32:a7:5a:1e    0.836s   (254) dc:a6:32:a7:5a:1e [     wlan0]
root@rp0:/home/tesis#
```

Fig. 14 Información de nodos

Para los nodos siguientes se aplicó la misma configuración, en cada uno de los nodos se configuró una dirección IP diferente de forma estática que estuvo dentro del rango de la red configurada en el módulo Raspberry del nodo servidor, estas configuraciones se observan en la (**Fig. 15**).

```
#!/bin/bash
sudo ifconfig wlan0 down
sudo iwconfig wlan0 mode ad-hoc
sudo iwconfig wlan0 channel 6
sudo iwconfig wlan0 essid 'tesisBlock'
sudo iwconfig wlan0 key 2350439861
sudo ifconfig wlan0 192.168.22.4 netmask 255.255.255.0 up
sudo modprobe batman-adv
sudo batctl if add wlan0
sudo ifconfig wlan0 up
sudo batctl gw_mode server
```

Fig. 15 Configuración de nodo dos

Después de configurar todos los nodos correspondientes con diferentes direcciones IP, en cada uno se verificó la conectividad entre nodos, esto se logró ejecutando el comando *ping* en todos los dispositivos, esto se observa en la (**Fig. 16**) y (**Fig. 17**).

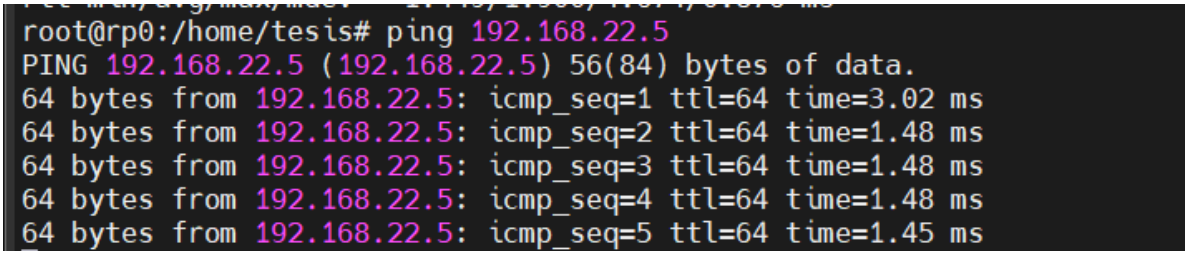


Fig. 16 Ping entre los nodos 0 al servidor

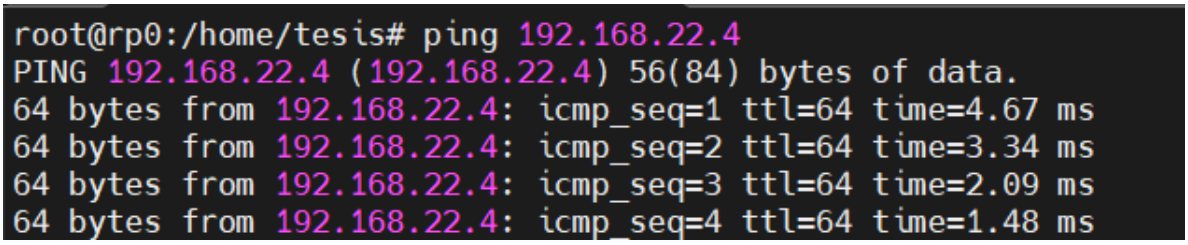


Fig. 17 Ping entre los nodos

4.5. Conexión Módulo PN532 con Raspberry pi modelo 4.

4.5.1 Integración del módulo PN532 RFID

En este apartado se describe la ejecución de la configuración de la conexión necesaria, entre el módulo PN532 con la Raspberry Pi modelo 4. En la (**Fig. 18**), se muestra cómo fue la conexión correcta para evitar problemas. El cable de color rojo representa el voltaje necesario para que el módulo PN532 funcione, por lo que se conectó a la Raspberry Pi modelo 4 a 5 voltios (V) mediante su pin VCC.

Por consiguiente, toda conexión se aterrizó para evitar que el módulo PN532 se dañe. Se conectó a tierra mediante su pin 6 (modelo Raspberry Pi modelo 4, al pin GND del módulo PN532. Ahora, el cable de color amarillo se conectó desde el pin SDA (Serial Data) del módulo PN532-pin 3, el cual se utilizó para enviar y recibir datos entre dispositivos utilizando el protocolo I2C.

Finalmente, el cable de color azul se conectó desde el pin SCL del módulo PN532 a la Raspberry Pi en el pin 5, que se utilizó para sincronizar la comunicación entre dispositivos utilizando el protocolo I2C. Este pin controla la velocidad a la que se envían y reciben los datos.

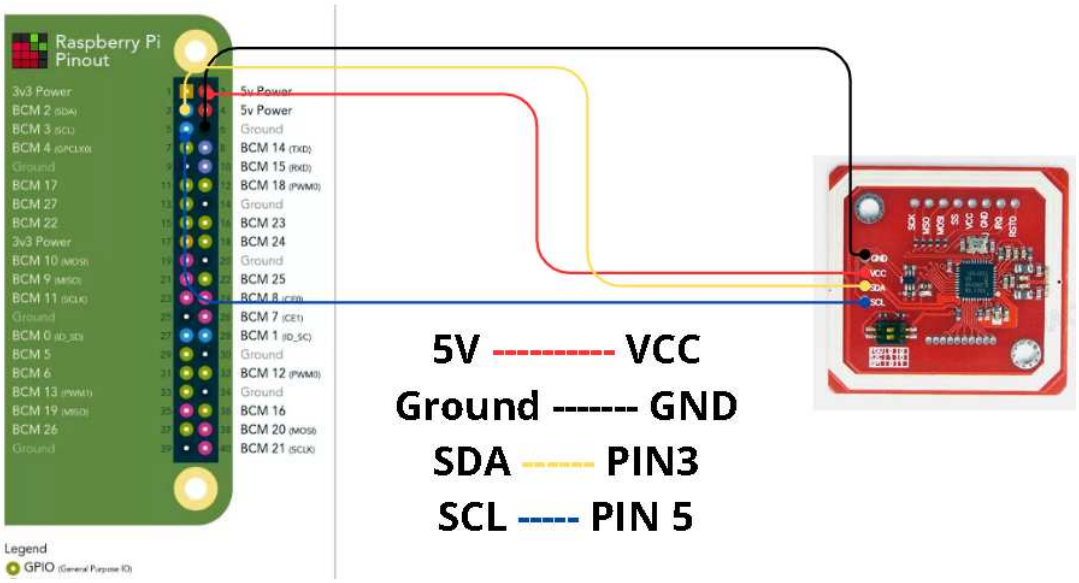


Fig. 18 Conexión módulo PN532 con Raspberry pi

4.6. Pruebas de lectura y envío del ID (Tag) RFID.

El funcionamiento del módulo RFID se ejecutó en conjunto con el Tag o tarjeta que cada estudiante de la universidad poseerá. El estudiante debe colocar el Tag/Tarjeta en el sensor de lectura del módulo RFID, el ID del Tag será transmitido al servidor Raspberry Pi y, posteriormente, a los demás nodos IoT, esto se visualiza en la (**Fig. 19**).

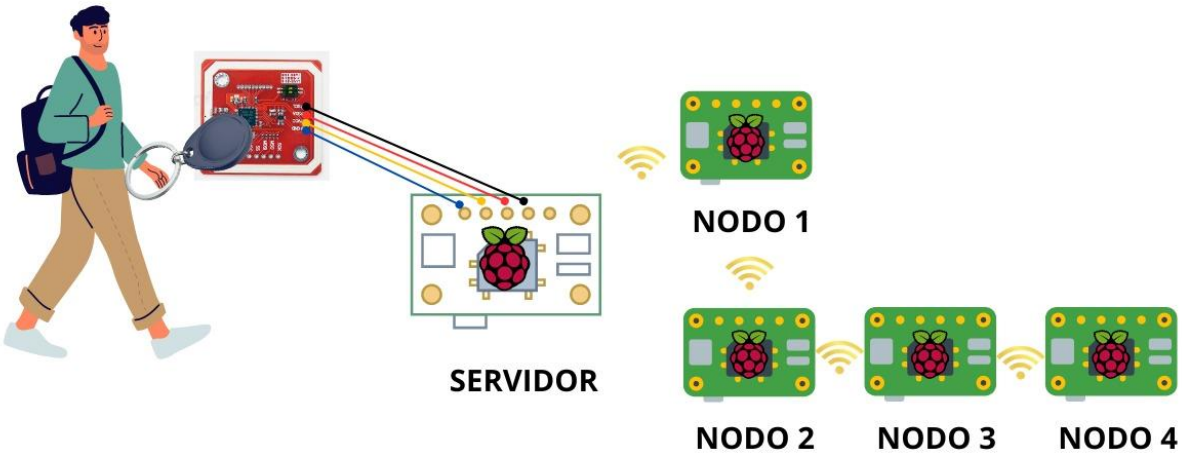


Fig. 19 Escaneo de Tag/Tarjetas RFID estudiante

En la (**Fig. 20**) se observa la conexión de todos los nodos Raspberry pi al módulo RFID y el proceso que se debe realizar en la vida de real para escanear el Tag o Tarjeta RFID para el ingreso correcto de los estudiantes.

Fig. 20 Escenario implementado de escaneo de Tag/Tarjetas RFID

Para la lectura de la tarjeta RFID, así como para el envío de los datos a los demás Raspberry pi, se utilizó una arquitectura cliente – servidor, utilizando sockets que permitieron la comunicación entre procesos que se ejecutaron en el mismo host o en diferentes hosts a través de la red. En la (**Fig. 21**) se muestra la referencia de la arquitectura propuesta.

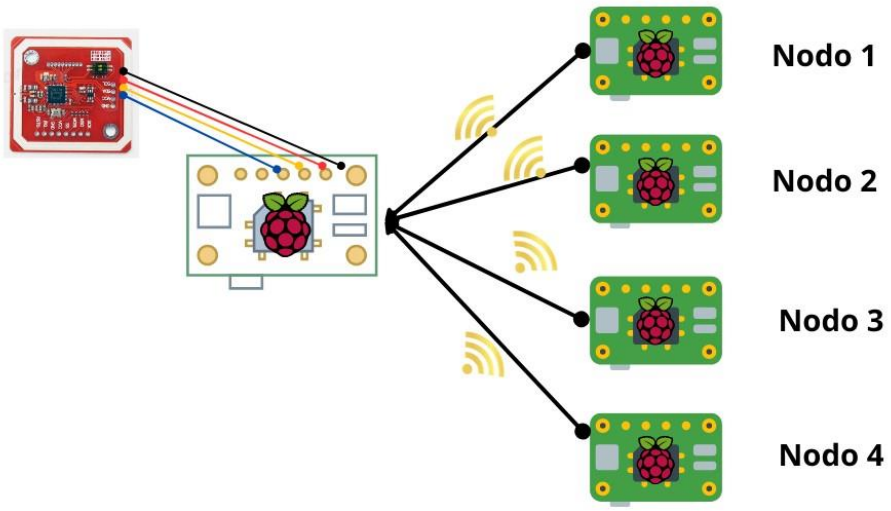


Fig. 21 Arquitectura implementada

Una vez definida la estructura, lo siguiente fue escribir el código correspondiente para ejecutar la lectura de la tarjeta/tag RFID, para esto, se usó el lenguaje de programación Python en el módulo Raspberry que hizo el papel de servidor.

El programa utilizó varias bibliotecas en Python para habilitar la funcionalidad deseada. En primer lugar, se importaron las bibliotecas *board* y *busio*, que proporcionaron abstracciones para interactuar con el hardware de la Raspberry Pi, incluyendo la configuración de pines y la comunicación a través de protocolos como I2C. Luego, se importó la biblioteca *DigitalInOut* del módulo *digitalio*, necesaria para trabajar con pines digitales de entrada/salida.

Además, se importó la clase *PN532_I2C* del módulo adafruit_pn532.i2c, que facilitó la comunicación con el lector de tarjetas PN532 a través del protocolo I2C. Por último, se importó la biblioteca socket para establecer conexiones de red TCP/IP y facilitar la comunicación con la máquina receptora. Estas bibliotecas nos permitieron interactuar con el hardware de la Raspberry Pi, leer los ID de tag/tarjetas y enviar la información de identificación a través de la conexión de red.

El código se centró en la lectura de tarjetas utilizando el módulo PN532, el cual se comunicó a través del protocolo I2C. En primer lugar, se inicializaron los objetos necesarios para la comunicación, como el bus I2C y el lector PN532. Se configuró el lector para que esté listo para detectar tarjetas RFID.

Luego, se estableció un bucle infinito que esperó por la detección del ID de una tarjeta. Una vez que se detectó una tarjeta, se leyó su UID y se convirtió a una cadena de texto. Posteriormente, se utilizó un socket TCP/IP para establecer una conexión con la máquina receptora, la cual se especificó mediante una dirección IP y un puerto. Finalmente, el UID de la tarjeta se envió a través de la conexión de red hacia la máquina receptora. Este código se ejecutó en el módulo Raspberry Pi que actuó como servidor en el contexto de este sistema de lectura de tarjetas, el código se podrá encontrar en el repositorio [20] de Github.

Una vez enviado, se necesitó receptar en los clientes, para ello se utilizó otro programa en Python capaz de leer los datos que venían desde el servidor. Este código en Python configuró un servidor TCP simple que escuchaba en una dirección IP y un puerto específico. Primero, se importó la biblioteca *socket*, que proporcionó las funciones necesarias para la comunicación a

través de sockets. A continuación, se definieron las constantes *HOST* y *PORT* para especificar la dirección IP y el puerto en los que el servidor escuchaba las conexiones entrantes.

Dentro de un bloque with, se creó un objeto de socket TCP utilizando *socket.socket()*, especificando *socket.AF_INET* para indicar el uso de IPv4 y *socket.SOCK_STREAM* para el tipo de socket TCP. Luego, se llamó a *bind()* para asociar el socket a la dirección y el puerto especificados.

Después de configurar el socket, se llamó a *listen()* para ponerlo en modo de escucha, permitiendo que acepte conexiones entrantes. Se imprimió un mensaje indicando que el servidor estaba escuchando en la dirección y puerto especificados.

Dentro de un bucle while True, se esperó y aceptó conexiones entrantes utilizando *accept()*, que devolvió un nuevo socket de conexión y la dirección del cliente *address*. Dentro de otro bloque with, se manejó la conexión *conn*. Se recibió el mensaje enviado por el cliente utilizando *recv()*, con un tamaño máximo de 1024 bytes. Luego, se imprimió el UID recibido decodificado a partir de los datos binarios. Finalmente, en la (**Fig. 22**) se muestra la ejecución del servidor en donde se muestra que la tarjeta fue detectada. Y en la (**Fig. 23**) se aprecia los datos recibidos en uno de los clientes.

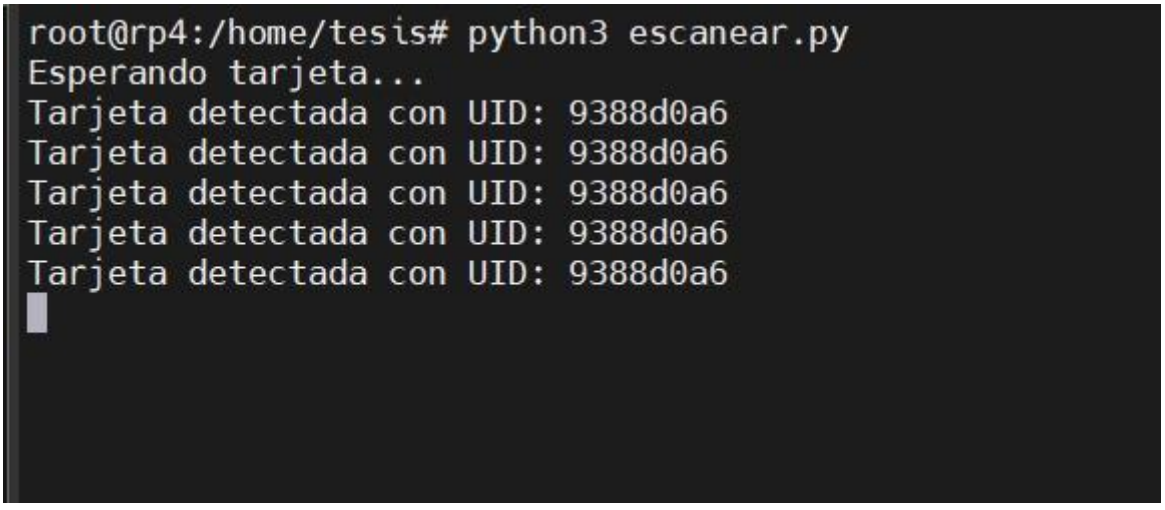


Fig. 22 Ejecución del servidor

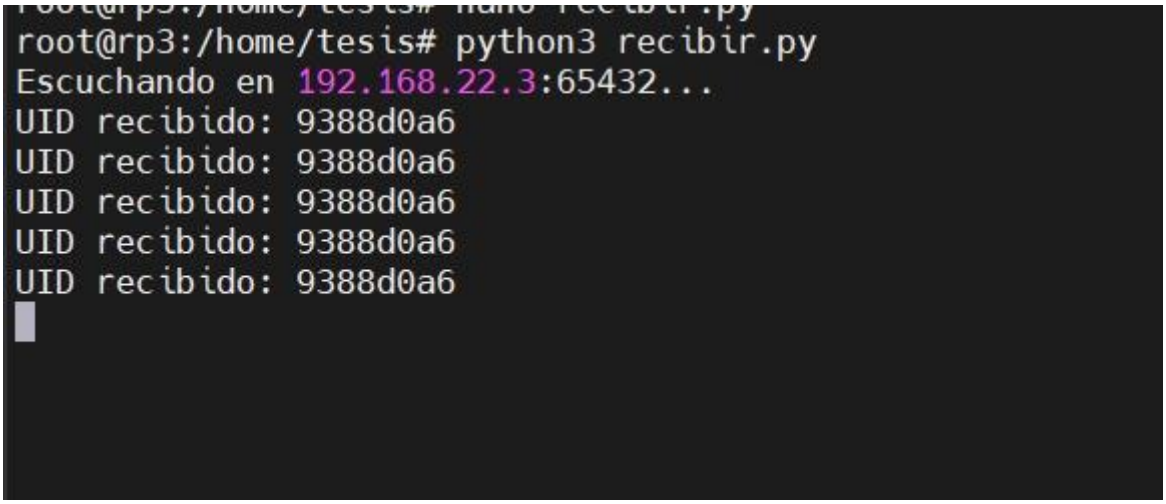


Fig. 23 Datos recibidos en uno de los clientes

4.7. Implementar encriptación usando blockchain.

Este código en Python estableció un servidor que utilizó un lector de tarjetas NFC (PN532) para recibir el ID de tag/tarjetas y construir una cadena de bloques basada en los datos de estas tarjetas. Se importaron las bibliotecas necesarias, incluyendo socket para la comunicación de red, busio para la comunicación I2C, y DigitalInOut y PN532_I2C para interactuar con el lector de tarjetas PN532.

Se definió una clase *Block* que representó un bloque en la cadena de bloques. Cada bloque contenía un hash del bloque anterior, datos, una altura opcional y una marca de tiempo. Se definió también una clase blockchain que contenía una lista de bloques y métodos para crear el bloque génesis y añadir nuevos bloques a la cadena.

Luego, se definió una función *hash_data()* que calculó el hash SHA-256 de los datos proporcionados. Seguido, se definió una función *send_block_to_next_client()* que envió un bloque (en forma de diccionario) al siguiente cliente a través de un socket. Fue necesario configurar el bus I2C y el lector PN532, y configurarlo para la detección de tarjetas. Se definió la dirección IP y el puerto del servidor para crear una instancia de la cadena de bloques.

Por último, se definió una función *handle_client()* que manejó la lógica de procesamiento de tarjetas para cada cliente conectado. Esta función leyó continuamente las tarjetas detectadas por el lector PN532, generó un hash para el UID de la tarjeta, agregó un nuevo bloque a la cadena de bloques con el hash y un "salt" único, y envió el bloque al cliente actual. Se creó el socket del servidor, se lo vinculó a la dirección IP y al puerto especificados, y se puso en modo de escucha. Finalmente se inició un bucle infinito que aceptaba conexiones de clientes entrantes. Para cada cliente aceptado, se creó un hilo de ejecución que manejó la conexión utilizando la función *handle_client()*.

En la siguiente (**Fig. 24**), se muestra el funcionamiento del código mencionado anteriormente.

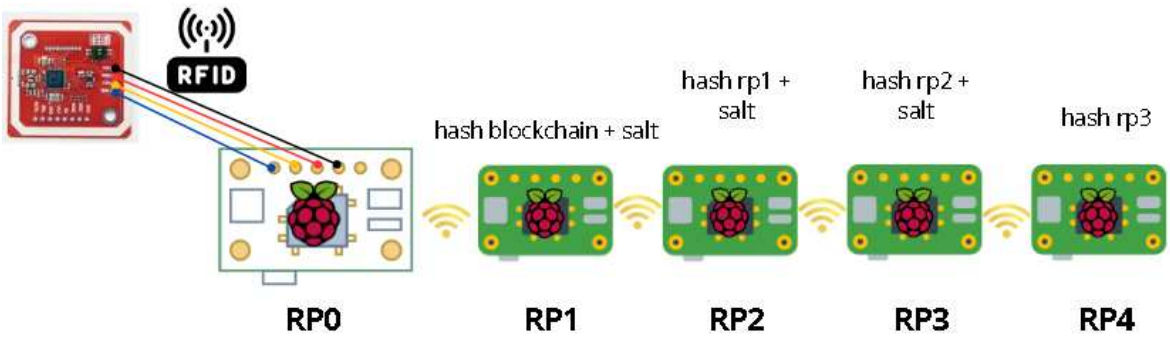


Fig. 24 Ilustración del funcionamiento del código

Siguiendo el mismo contexto, se implementó un cliente que se conectó a un servidor y recibió bloques de datos en formato JSON. Se importaron las bibliotecas necesarias, incluyendo socket para la comunicación de red, *json* para manipular datos en formato JSON, *hashlib* para calcular hashes y *os* para generar un nuevo "salt". Se definió una función *hash_data()* para calcular el hash SHA-256 de los datos proporcionados y también se configuraron las direcciones IP y los puertos del servidor y del siguiente cliente (RP2 – Raspberry PI 2). Se creó un socket TCP para el cliente, se estableció una conexión con el servidor y se imprimió un mensaje indicando que se ha establecido la conexión con el servidor.

Luego, si la clave 'salt' estaba presente, se generaba un nuevo "salt" único utilizando la función *os.urandom(16).hex()* y se calculó un nuevo hash combinado usando el hash anterior y el nuevo "salt". Se imprimió el nuevo hash combinado y se creó un nuevo diccionario con el nuevo hash combinado y el nuevo "salt". Finalmente, se envió el nuevo bloque al siguiente cliente en formato JSON y si la clave 'salt' no estaba presente en los datos recibidos, se imprimía un mensaje de error.

4.8. Ethereum Aplicación Descentralizada.

Una aplicación descentralizada o *(dApp)* es un tipo de aplicación informática que opera en una red descentralizada, lo que significa que no depende de un servidor central para funcionar. En lugar de eso, la aplicación se ejecutó en múltiples nodos de la red, y las decisiones y operaciones se tomaron de manera distribuida entre estos nodos en lugar de ser controladas por una única entidad central.

Para poner un ejemplo de las aplicaciones tradicionales, se ejecutaron en un solo servidor central, la aplicación descentralizada en cambio se ejecutó en una red formada por

miles de nodos u ordenadores. En la (**Fig. 25**) se muestra el diseño de una aplicación tradicional con la diferencia de una aplicación descentralizada.

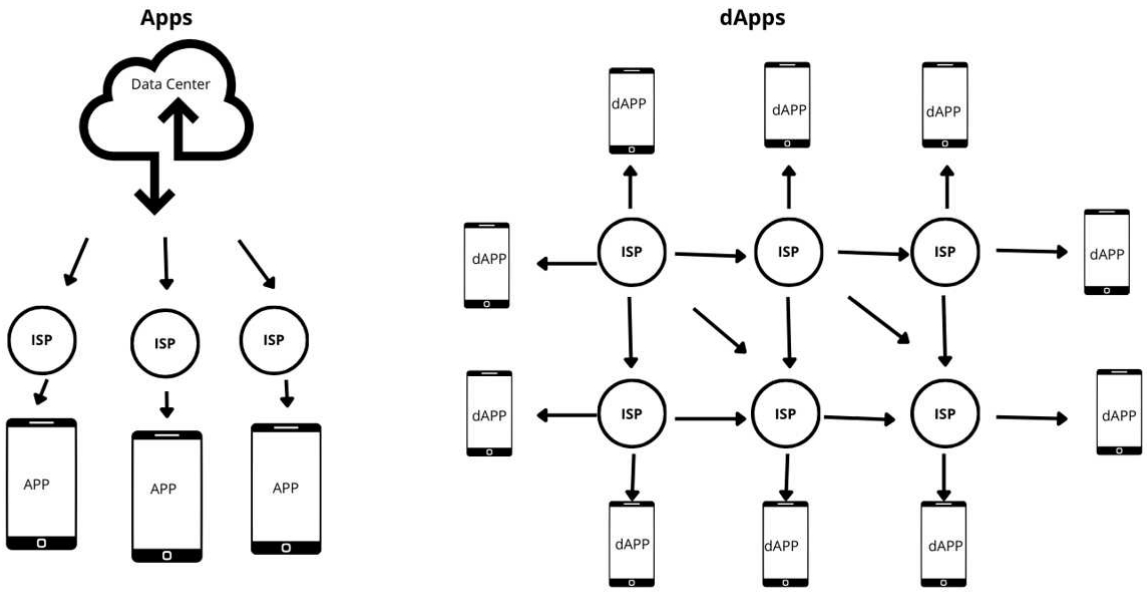


Fig. 25 Ilustración en donde se aprecia la diferencia de la estructura de aplicaciones tradicionales vs aplicaciones descentralizadas

Para comprender la diferencia entre una aplicación tradicional y una aplicación descentralizada, debemos entender sus diferencias en dos niveles importantes, el backend y el almacenamiento de datos. En una aplicación tradicional, el backend, es la parte lógica de las aplicaciones que la hace funcionar dinámicamente, esto se almacena en el ordenador o servidor central. Sin embargo, en las aplicaciones descentralizadas, el backend está implementado en un smart contract(contrato inteligente) que se ejecuta sobre una Blockchain como Etherium. Cuando se utilizan contratos inteligentes como backend estos actúan como la capa de lógica empresarial.

En lugar de depender de un servidor centralizado para manejar la lógica y la ejecución de las operaciones, las aplicaciones pueden utilizar contratos inteligentes para gestionar las transacciones, validar las condiciones y ejecutar automáticamente las acciones acordadas cuando se cumplen ciertas condiciones. Una vez que se implementa un contrato inteligente en una cadena de bloques como Ethereum, se convierte en una parte integral de esa cadena de bloques y se distribuye a todos los nodos de la red. Garantiza que todas las transacciones y operaciones relacionadas con el contrato queden registradas permanentemente en la cadena de bloques, garantizando transparencia y seguridad.

El siguiente elemento diferenciador es el proceso de almacenamiento de datos. En las aplicaciones tradicionales los datos son almacenados en un computador u ordenador central lo

que puede ser vulnerado de múltiples maneras. Esto no ocurre con las aplicaciones descentralizadas en las que los datos quedan almacenados en la red de blockchain, de esta forma se garantiza que nunca se pierdan los datos del usuario.

En una aplicación tradicional, los datos suelen almacenarse en servidores centralizados controlados por una entidad o empresa específica. Esta entidad es responsable de mantener y proteger los datos, y los usuarios de la aplicación acceden a ellos a través de una interfaz de usuario centralizada. En las aplicaciones descentralizadas, la distribución de datos en una red descentralizada puede hacer que sea más difícil para los atacantes comprometer o censurar los datos. Además, las tecnologías de blockchain pueden proporcionar inmutabilidad y transparencia a los datos almacenados en la cadena de bloques.

4.9. Implementación de aplicativo web descentralizado para control de acceso.

Para la implementación de una página web descentralizada fueron necesarios instalar algunos programas. Primero se configuró un entorno en tiempo de ejecución multiplataforma de código abierto que permitió la iniciación de un servidor web, por lo que fue necesario descargar *node js* desde sus repositorios, para no tener problemas los comandos los puede encontrar en el enlace a GitHub que se mencionó en el apartado anterior. Por otro lado, fué importante la instalación de *ganache cli*. Ganache CLI es una herramienta de línea de comandos que proporciona una cadena de bloques Ethereum local para el desarrollo, con el comando *npm install -g ganache-cli* se pudo instalar de forma global. El siguiente requerimiento, fue Truffle, que es un framework de desarrollo para Ethereum que facilita la creación, el despliegue y las pruebas de contratos inteligentes, se instaló con el comando *npm install -g truffle*.

4.9.1. Preparación entorno de trabajo.

Una vez que se ha descargado lo necesario, el siguiente paso fue inicializar un proyecto con el comando *truffle init <nombreproyecto>* esto creará una carpeta en el directorio que se le haya especificado. Es importante entender la estructura del proyecto, una vez dentro de la carpeta nos encontraremos con el archivo *trufflle-conFigura.js,* este fue el encargado de conectar nuestro aplicativo en la red blockchain. Luego, tenemos la carpeta *contracts* dentro del archivo es donde se configuró todo nuestro contrato, que fueron básicamente funciones que se ejecutaron en la blockchain. Luego, se dispuso del fichero *migrations* que contiene el archivo necesario para subir la aplicación descentralizada a la blockchain. A continuación, en la (**Fig. 26**) se muestra la estructura general del proyecto.

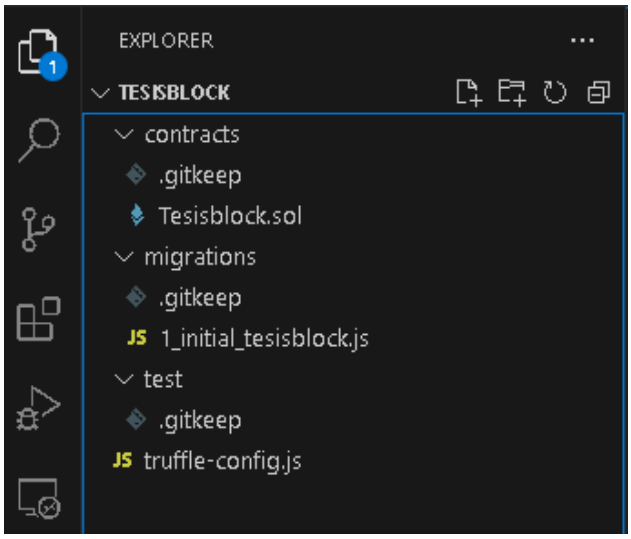


Fig. 26 Estructura del proyecto inicial Solidity

4.9.2. Creación de un contrato inteligente.

El contrato comienza con la declaración de la versión de Solidity que se utilizará y la especificación de la licencia bajo la que se distribuye el contrato. Luego, se define la estructura Task, que representa una tarea y contiene varios campos como *id, idTarjeta, description, done, createdAt, nombres, apellidos, cedula, celular, email y carrera*. Cada tarea tiene un identificador único (id) y un estado que indica si la tarea está completada o no (done). El contrato también define tres eventos: TaskCreated, TaskToggledDone y TaskEdited. Estos eventos se utilizaron para notificar a los clientes cuando se haya creado una nueva tarea, cuando se haya cambiado el estado de una tarea (completada o no) y cuando se haya editado una tarea, respectivamente.

Además de la estructura y los eventos, el contrato también incluye una variable *tasksCounter* para realizar un seguimiento del número total de tareas creadas y un mapeo *tasks* que asocia el identificador de cada tarea con su estructura correspondiente.

La función *createTask* se utilizó para crear una nueva tarea y almacenarla en el mapeo de *tasks*. Esta función incrementó *tasksCounter* para asignar un nuevo identificador único a la tarea creada y emitir el evento *TaskCreated* para notificar la creación de la tarea. Luego, la función *toggleDone* permitió cambiar el estado de una tarea entre completada y no completada. Recibió como parámetro el identificador de la tarea y actualizó el estado de la tarea correspondiente en el mapeo de tasks. Luego, emitió el evento *TaskToggledDone* para notificar el cambio de estado. Finalmente, la función *editTask* se utilizó para editar los detalles de una tarea existente. Recibe como parámetros el identificador de la tarea y los nuevos detalles de la

tarea. Verifica si la tarea existe y, si es así, actualiza los campos correspondientes de la tarea en el mapeo *tasks*. Luego, emitió el evento *TaskEdited* para notificar la edición de la tarea.

4.9.3. Ejecución de Ganache-CLI

Ganache-cli nos permitió ejecutar una red con la que la blockchain va a ser ejecutada, con el comando *ganache-cli --host 192.168.7.6 --port 7545* se especificó el host cuya IP corresponde a la máquina en donde se está ejecutando el ganache-cli, y el puerto por defecto es el 7545, que fue el encargado de la comunicación con el metamask. Cuando se ejecutó ganache mostró una serie de parámetros en donde lo importante son las cuentas disponibles junto con sus direcciones y saldos. Cada cuenta tiene una dirección y un saldo de 100 ETH. Seguido, se mostraron las claves privadas correspondientes a cada una de las cuentas listadas. Estas claves privadas son necesarias para firmar transacciones en la red Ethereum.

En la (**Fig. 27**) se aprecian algunas cuentas locales, así como sus claves privadas para poder vincularse con la cartera electrónica de metamask, que es un dinero ficticio que nos ofrece ganache para poder ejecutar.

```
root@rp4:/home/tesis# ganache-cli --host 192.168.7.6 --port 7545
Ganache CLI v6.12.2 (ganache-core: 2.13.2)
eth_blockNumber

Available Accounts
==================
(0) 0x7B1D510A42FECA4A3F7db1C8Ca93BA1e576e51C7 (100 ETH)
(1) 0x5Ec54D89a42E1b27b7C12151fAd061BB0Ef6B423 (100 ETH)
(2) 0x6B36f1a057fc1AFDd3068af293863335B4E17E27 (100 ETH)
(3) 0xE5010B3a04bA94984aAb59f8305F3EF7Eb27e9E7 (100 ETH)
(4) 0x8F8C409dd17C97019d134D68f2cfC95b9328aAd7 (100 ETH)
(5) 0xfD4d220075965810273342e2Ca2CD7719803157D4 (100 ETH)
(6) 0x08d1FE79AbE081e994C1921F68307B8973503C37 (100 ETH)
(7) 0x0b9a5eEeBF5af719100c03D635503E8D64E663A9 (100 ETH)
(8) 0x84410c61995701BE512f2B16162f8402AfEdDa7c (100 ETH)
(9) 0x9f6f950bB5944D7C25843EdE7CDF291d55F45082 (100 ETH)

Private Keys
==================
(0) 0x4f064d13465e2fde000bdea7888dc464f9aa452881ff322b45ab5ced06387c67
(1) 0x8df8a589481c00077c7f8e9d30201aabf77349ee70976c8024db2d3bec457487
(2) 0x7d8dcf2f38aa27a665c340b79dfea6ee862f9f39378d88e55b5f1845a8fa91e4
(3) 0x5857c172729dda0f8f14ccc095991895d0e0346105e654b2ffce37fbd7030ee2
(4) 0x9c08686e3c2255be0670097211a42d1c8d73275e40cb872afa3805c0bcc84071
(5) 0x17f7c0200546986500370a44ce4ceb88eff931ee0dfcad14ad894449cdf4143d
(6) 0xa7fc26f8514c6926d1f3c7165143d18565fe6922bba545e406f3fa03c6fad9ea
(7) 0x5847f3c807083c110727d7551ae3d2fea0341b6687baa7695fd9015e51cf628c
(8) 0x56e7c5b4153e61f2da18b2e189bf47795108ee1ccbbaf84aed90c80168f32526
(9) 0xb55b07d492d42bdceacaa4bfb24c164caa1304a73ed1b60346488e5bdb03453e

HD Wallet
==================
Mnemonic:      produce answer alpha govern priority nothing enough mushroom denial please castle explain
Base HD Path:  m/44'/60'/0'/0/{account_index}
```

Fig. 27 Cuentas y claves privadas de ganache-cli.

Fue necesario generar la carpeta *build* de nuestro contrato, para ello, con el comando *troffle compile* se hizo una compilación del proyecto que contendrá los archivos generados durante el proceso de compilación. El archivo más importante fue el denominado ABI, este archivo contiene la definición de la interfaz de un contrato inteligente en formato JSON. El ABI

describe los métodos y eventos disponibles en tu contrato y cómo interactuar con ellos desde el exterior.

El siguiente elemento es el denominado *bytecode* que es el código del contrato inteligente compilado en formato hexadecimal. Es el código que se ejecutará en la cadena de bloques Ethereum. Por último, el JSON del contrato desplegado guarda información sobre el contrato una vez que ha sido desplegado en la red Ethereum. Contiene la dirección del contrato desplegado y otros detalles relevantes.

4.9.4. Creación Interfaz web.

Para implementar la interfaz web se utilizaron HTML y JavaScript, que estuvieron dentro de la comunicación con nuestro contrato inteligente. Creamos un nuevo directorio denominado *client* que tuvieron cuatro archivos principales que se muestra en la (**Fig. 28**).

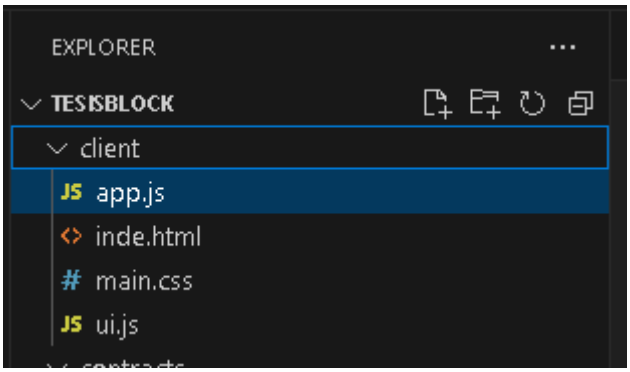


Fig. 28 Estructura de interfaz de usuario.

Fue necesario tener un servidor que permita la interacción entre el backend (smart contracts) con el frontend. Para la consola se instaló un módulo de node que nos permitió tener un servidor web integrado. Primero se instaló el módulo de *lite-server* que permitió recargar la página después de algún cambio automáticamente con el comando *npm i lite-server Bootstrap*, para los estilos de la aplicación se instaló el framework Bootstrap, que dió la facilidad de tener mejores estilos personalizados. Finalmente, después de la ejecución del comando, se creará una carpeta denominada *node_modules* que tuvo todo lo necesario para que nuestra aplicación funcione correctamente.

En este punto, fue necesario realizar una configuración personalizada que nos permita la ejecución y configuración del servidor. Por lo que se creó una carpeta denominada *bs-config.js* este archivo configura un servidor web utilizando BrowserSync, donde se especifican los directorios base desde los cuales se servirán archivos estáticos (/client y ./build/contracts), junto con una ruta específica ("/libs") que servirá archivos desde ./node_modules. El servidor

estará en el puerto 3000, facilitando el desarrollo y la visualización de aplicaciones web en tiempo real con recarga automática de los navegadores.

4.9.5. Programación index.html

A continuación, se explica la estructura del código HTML, en la (**Fig. 29**) se muestra cómo se visualiza en el navegador.

1. Se inició con la declaración del tipo de documento (<!DOCTYPE html>) seguido de la etiqueta raíz <html> que especifica el idioma (lang="en").
2. Contiene metadatos como la codificación (<meta charset="UTF-8">), la compatibilidad con IE (<meta http-equiv="X-UA-Compatible" content="IE=edge">), la configuración de la vista del dispositivo (<meta name="viewport" content="width=device-width, initial-scale=1.0">), el título de la página (<title>Control Sistema ESPE</title>), enlaces a archivos de estilo CSS y fuentes externas.
3. Contiene el contenido visible de la página. En este caso, incluyó una estructura de contenedores ( ) y elementos de formulario (<form>) para registrar nuevos estudiantes. También hay una tabla (<table>) para mostrar los estudiantes registrados, aunque inicialmente está vacía (<tbody id="tasksTableBody">). Además, se incluyeron enlaces a scripts JavaScript al final del cuerpo para manejar la lógica de la aplicación y la interacción del usuario.

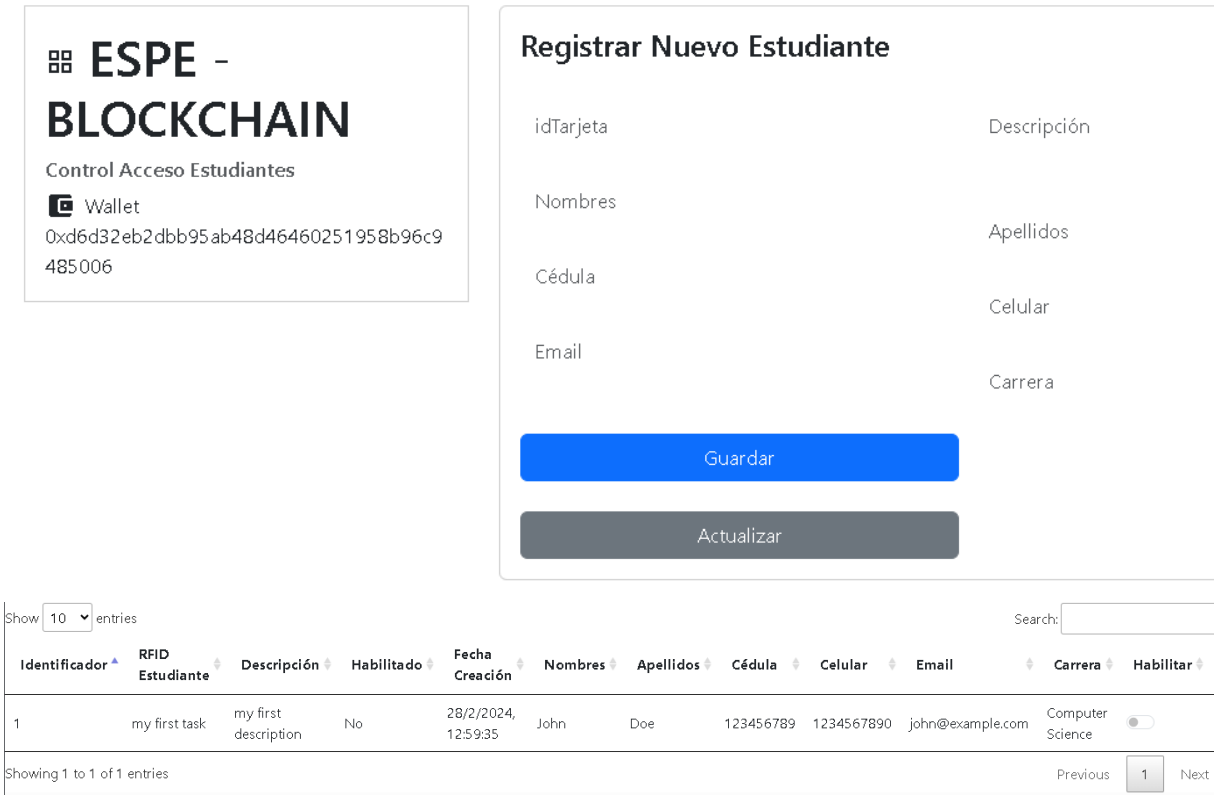


Fig. 29 Interfaz principal del listado de estudiantes

4.9.6. Programación app.js

Se constituyó la lógica de una aplicación web diseñada para interactuar con un contrato inteligente en la blockchain Ethereum, enfocándose en el control de acceso de estudiantes. El objeto App actúa como el núcleo de la aplicación, gestionando las diferentes funcionalidades y la interacción con la cadena de bloques. Inicialmente, el método init se encargó de iniciar la aplicación, asegurándose de cargar correctamente la conexión a Web3, la cuenta del usuario y el contrato inteligente antes de renderizar la interfaz de usuario y las tareas asociadas.

El método *loadWeb3* verificó la disponibilidad de una instancia de MetaMask o una extensión similar, estableciendo la conexión adecuada para interactuar con la blockchain. Por su parte, loadAccount solicita al usuario su cuenta Ethereum y la almacena para futuras operaciones. El método loadContract carga el contrato inteligente desde un archivo JSON, lo instancia y configura su proveedor de Web3, permitiendo así su uso dentro de la aplicación.

Una vez inicializada la aplicación, el método render actualiza la interfaz con la dirección de la cuenta del usuario, mientras que renderTasks recupera las tareas del contrato inteligente y las muestra en una tabla HTML, permitiendo su visualización y edición. El método createTask se encargó de crear nuevas tareas en el contrato inteligente, mientras que toggleDone permitió marcar una tarea como completada o no completada, actualizando su estado en la blockchain.

El método toggleDone es esencial para cambiar el estado de una tarea entre completada y no completada dentro de la aplicación. Funciona mediante la interacción con un checkbox asociado a cada tarea en la interfaz de usuario. Al activarse este checkbox, se ejecuta la función toggleDone, que extrae el identificador único de la tarea desde los datos asociados al elemento HTML. Luego, utiliza este identificador para llamar a la función correspondiente en el contrato inteligente, toggleDone, pasando como argumentos el ID de la tarea y la cuenta del usuario que realiza la acción. Una vez que la transacción se completa en la blockchain, se actualiza la página para reflejar los cambios en el estado de la tarea.

Por otro lado, el método getTaskByIdTarjeta se encargó de obtener los detalles de una tarea específica según su identificador de tarjeta. Este identificador se utiliza para acceder a la tarea deseada en el contrato inteligente. La función extrae los detalles de la tarea, como su título, descripción, estado, fecha de creación y datos del estudiante asociado, incluyendo nombres, apellidos, cédula, celular, email y carrera. Luego, utiliza estos datos para generar una estructura HTML que representa la tarea, la cual se inserta en un elemento específico dentro de la interfaz de usuario, identificado por el ID tasksList.

Finalmente, el método editTask permitió modificar los detalles de una tarea existente en la blockchain Ethereum. Recibe como parámetros el ID de la tarea a editar y los nuevos

detalles que se desean aplicar, como una nueva descripción, nombres, apellidos, etc. Utiliza estos datos para llamar a la función editTask en el contrato inteligente, pasando los argumentos necesarios. En caso de que la operación se complete con éxito, se imprime un mensaje en la consola indicando que la tarea se ha editado correctamente. Sin embargo, si surge algún error durante el proceso de edición, se muestra una alerta al usuario notificando que algo salió mal. De esta manera, este método facilita la gestión dinámica y la actualización de las tareas dentro de la aplicación.

4.9.7. Programación ui.js

Este código establece un evento que se activa cuando el DOM ha sido completamente cargado. Una vez que esto sucede, se llama al método App.init(), el cual inicializa la aplicación. Luego, se obtiene una referencia al formulario de tareas mediante document.querySelector("#taskForm"). Se añade un evento de escucha al formulario que se activa cuando se envía (submit). Cuando el formulario se envía, se previene el comportamiento por defecto (la recarga de la página). A continuación, se obtienen los valores de los campos del formulario, incluyendo el ID de la tarjeta y la descripción de la tarea. Se añaden los demás campos necesarios para crear una tarea, como nombres, apellidos, cédula, celular, email y carrera. Luego, se intenta crear una nueva tarea llamando al método App.createTask(), pasando los valores obtenidos de los campos del formulario como argumentos. Si ocurre algún error durante la creación de la tarea, se imprime en la consola y se muestra una alerta al usuario indicando que algo salió mal. En resumen, este código facilita la creación de nuevas tareas dentro de la aplicación cuando se envía el formulario correspondiente.

4.9.8 Lector de tarjeta RFID

Este código utiliza Flask, una biblioteca de Python para construir aplicaciones web, junto con otras librerías para interactuar con un lector de tarjetas RFID. Primero, se importan las bibliotecas necesarias, como Flask para la aplicación web, busio y board para configurar el bus I2C, y DigitalInOut y PN532_I2C de adafruit_pn532 para interactuar con el lector de tarjetas PN532 a través del protocolo I2C. Además, se importa CORS para manejar las solicitudes de recursos cruzados (CORS) en la aplicación web. Se crea una instancia de la aplicación Flask y se habilita CORS para permitir solicitudes desde diferentes dominios. Luego, se configura el bus I2C y se inicializa el lector de tarjetas PN532 para la tarjeta.

Se define una ruta *get_rfid* que acepta solicitudes GET. Cuando se realiza una solicitud a esta ruta, el código espera hasta que se detecta una tarjeta RFID. Si se detecta una tarjeta, se lee su identificador y se convierte a una cadena de texto hexadecimal. Luego, se devuelve este identificador en formato JSON como respuesta a la solicitud.

Finalmente, se ejecuta la aplicación Flask en el host '192.168.7.6' y el puerto 5000 en modo de depuración. Esto significa que la aplicación estará disponible en la dirección IP especificada y escuchará las solicitudes en el puerto 5000, además de mostrar mensajes de depuración en la consola en caso de errores.

4.9.9. Mostrar ID tarjeta en el campo del HTML.

Este código en JavaScript se encargó de realizar solicitudes periódicas a una API RESTful ubicada en la dirección URL 'http://192.168.7.6:5000/get_rfid' para obtener el ID de una tarjeta RFID. La variable permitirNuevoEscaneo controla si se permite realizar un nuevo escaneo en un momento dado, evitando escaneos consecutivos indeseados.

La función obtenerIDTarjeta() se encarga de realizar la solicitud a la API si se permite un nuevo escaneo. Cuando se recibe una respuesta exitosa, extrae el ID de la tarjeta de los datos recibidos y lo muestra en un campo de entrada en la interfaz de usuario. Luego, se establece un temporizador para permitir un nuevo escaneo después de un período de 5 segundos. La función mostrarIDTarjeta(idTarjeta) se utiliza para mostrar el ID de la tarjeta en un campo de entrada específico de la interfaz de usuario.

Además, se establece un intervalo de 1 segundo utilizando setInterval() para llamar a la función obtenerIDTarjeta() repetidamente cada segundo, lo que permite mantener actualizado el ID de la tarjeta en la interfaz de usuario en tiempo real. Esto se puede ajustar según sea necesario cambiando el valor pasado al setInterval().

4.9.10. Configuración Metamask.

Ahora necesitamos ejecutar nuestra aplicación en la blockchain, por lo que vamos a ocupar una extensión del navegador denominada metamask. Luego de que se instala la extensión, deberemos seguir los siguientes pasos para configurar de una manera correcta.

1. Aceptar acuerdos y seleccionar la opción de crear una nueva cartera (**Fig. 30**).

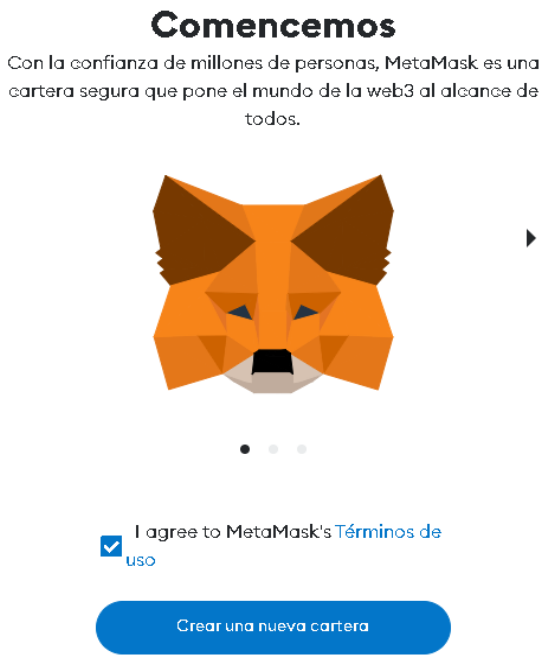


Fig. 30 Acuerdos de Metamask

2. Configurar una nueva contraseña (**Fig. 31**)

Fig. 31 Contraseña de metamask

3. Seleccionar la opción de asegurar mi cartera (**Fig. 32**), lo que mostrará una opción de recuperación. Metamask no usa una recuperación por correo electrónico, si no que con unas frases de recuperación.

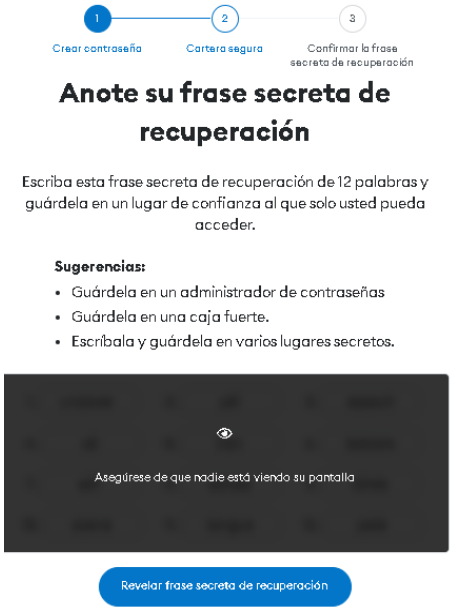


Fig. 32 Frase secreta de recuperación

4. Una vez que se revela la frase de recuperación, lo siguiente es verificar esa frase (**Fig. 33**).

Fig. 33 Configuración frase de recuperación

5. Finalmente, se logra la conexión a la cartera (**Fig. 34**).

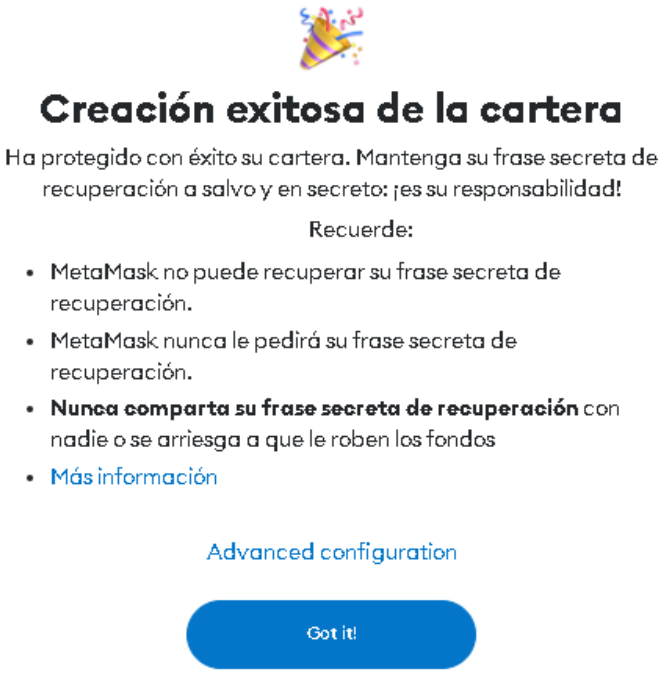


Fig. 34 Creación exitosa de la cartera.

4.9.11. Configuración de red Etherium.

1. Para configurar una red Etherium en nuestra cartera de Metamask. Debemos dirigirnos a la configuración (**Fig. 35**).

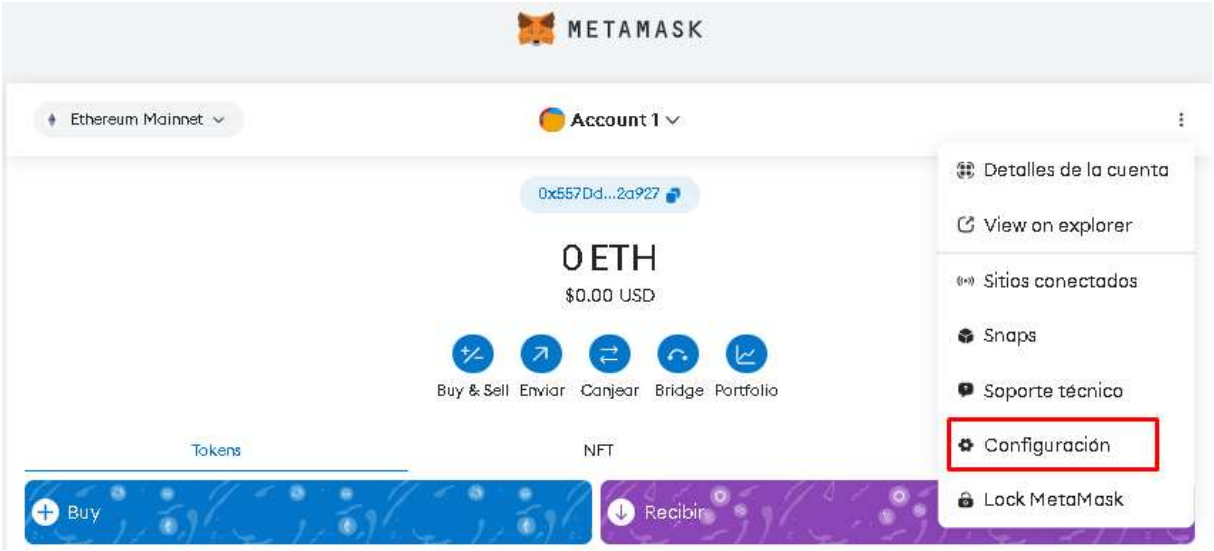


Fig. 35 Ir a configuración metamask

2. Se busca la opción que diga redes para posteriormente darle en la opción de agregar una nueva red (**Fig. 36**).

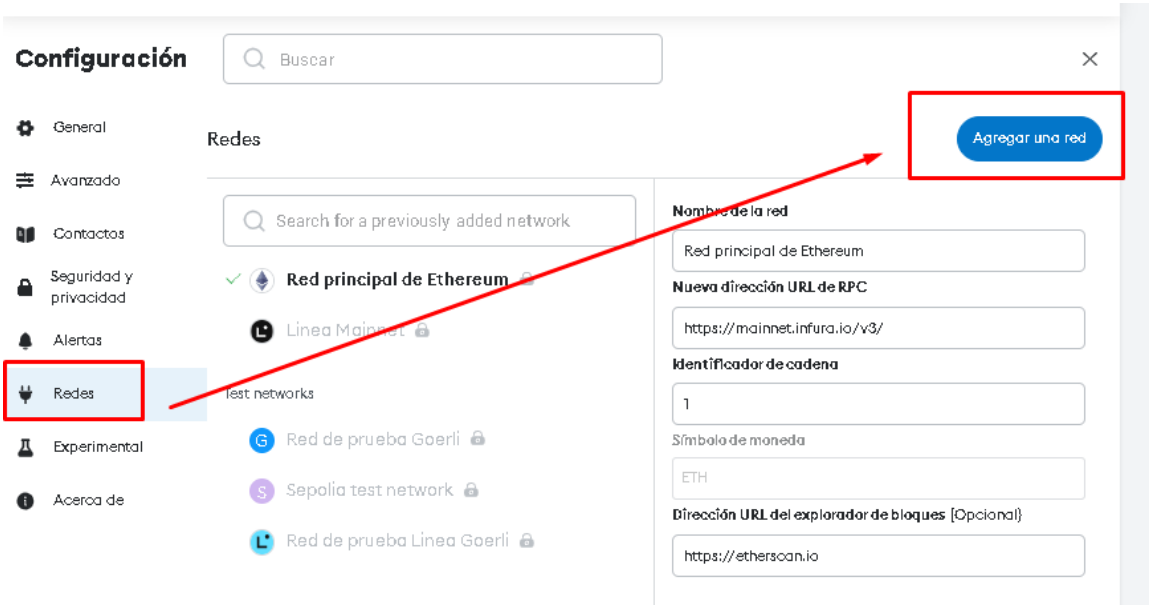


Fig. 36 Agregar una red

3. Buscamos la opción de agregar una red manualmente (**Fig. 37**)

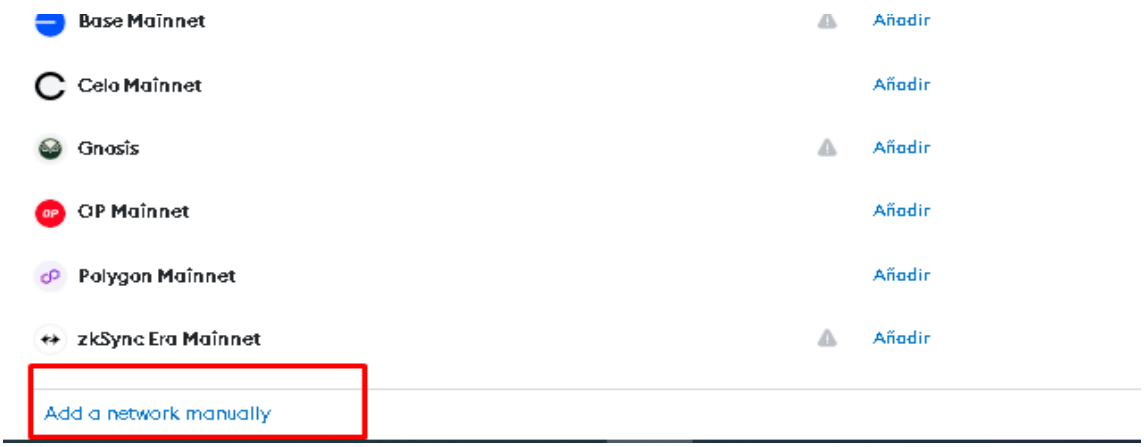


Fig. 37 Agregar una nueva red manualmente

4. Ahora se procede a configurar los parámetros necesarios (**Fig. 38**), donde el nombre puede ser cualquier nombre en nuestro caso redprivadatesis. El segundo parámetro es donde se está ejecutando el servidor de ganache seguido del puerto que por lo general

es el 7545. Finalmente, se selecciona el tipo de moneda que es el ETHER por su abreviatura ETH.

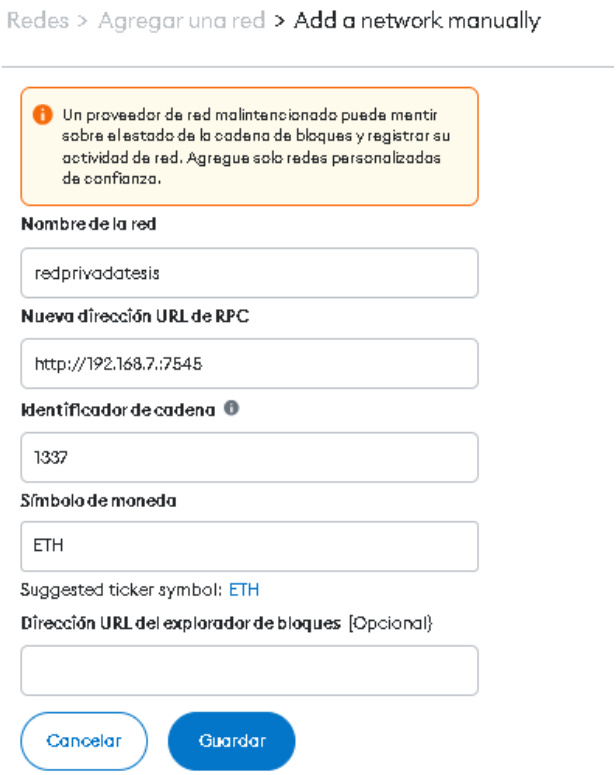
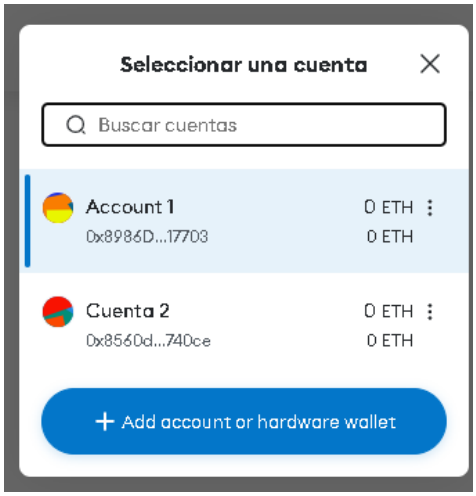
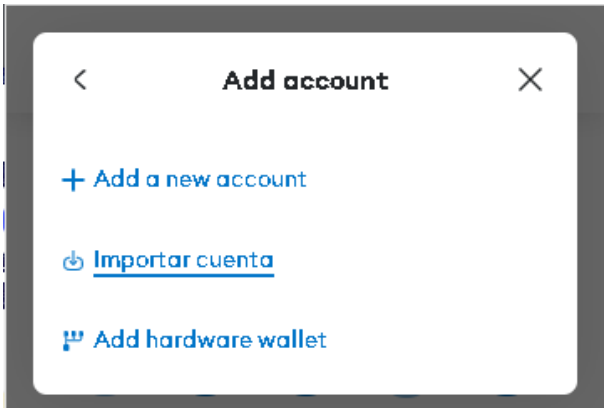


Fig. 38 Configuración de parámetros para la nueva red

Ahora, con las claves privadas que proporciona ganache escogemos una y la importamos dentro de nuestra red ya creada mire la (**Fig. 39**).





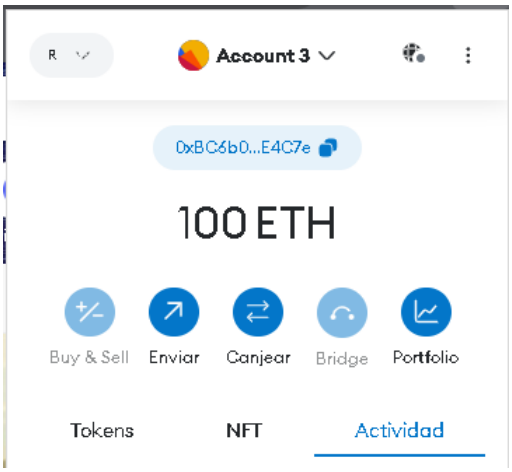


Fig. 39 Importar clave privada en la cartera de Etherium.

Por el momento cuando ingresamos la clave privada nos dará 100 ETH. Lo siguiente es ejecutar, el servidor web si antes primero haber desplegado el proyecto, cuando se hace el deploy se está uniendo la blockchain a la red ya creada. Cuando accedes a la página web nos pedirá realizar la conexión con Metamask (**Fig. 40**).

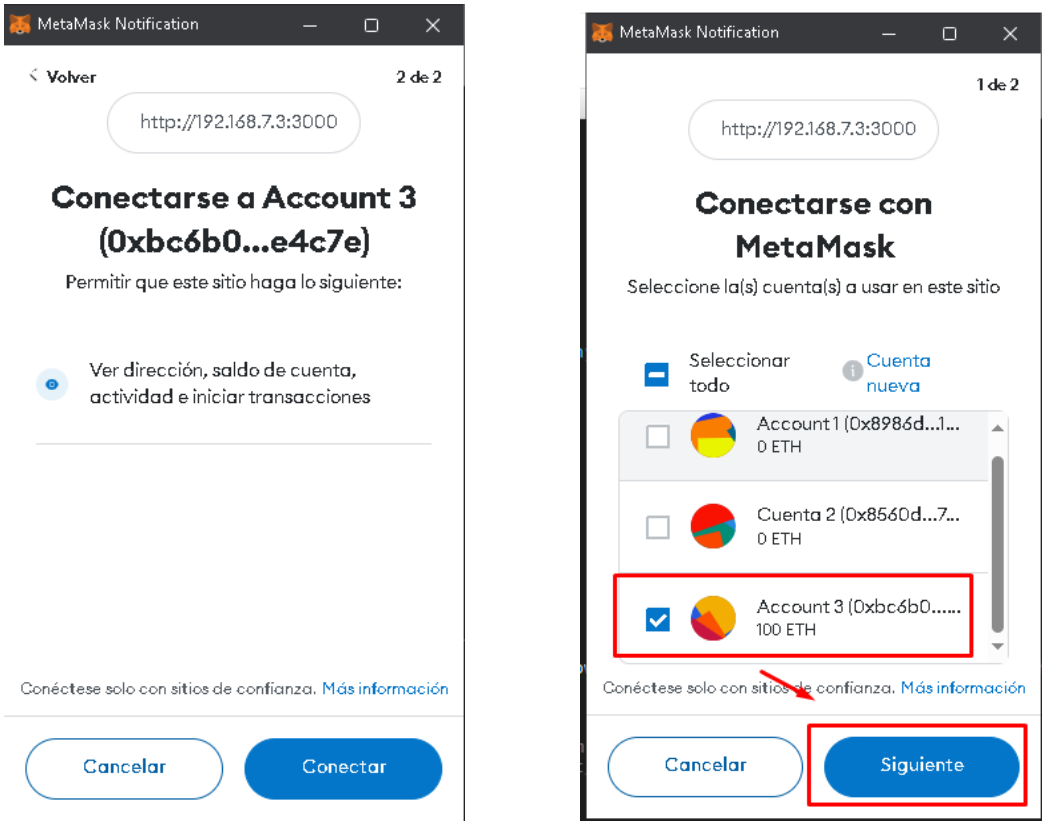


Fig. 40 Conexión del navegador web con Metamask

Si la conexión se logra exitosamente, nuestra cartera cambiará de saldo, mire la (**Fig. 41**).

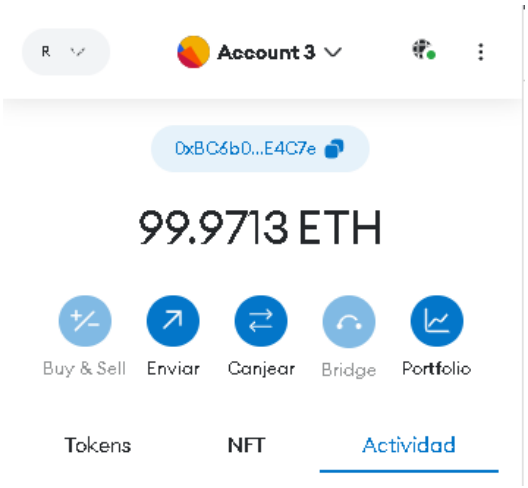


Fig. 41 Dinero que nos ofrece ganache-cli para la blockchain.

4.10. Nodos a la red Blockchain.

Añadir nodos a la red blockchain significa que vamos a agregar más dispositivos Raspberry pi o dispositivos que estén ejecutando software de nodo blockchain y estén conectados a la red existente. En un sistema blockchain, un "nodo" es básicamente un participante en la red que mantiene una copia de blockchain completa y participa en el proceso de validar y registrar transacciones. En la (**Fig. 42**) se muestra la estructura completa al momento de añadir los nodos. Estos nodos ayudan a mantener segura la cadena de bloques, entre más nodos estén conectados mayor seguridad tendrá la blockchain.

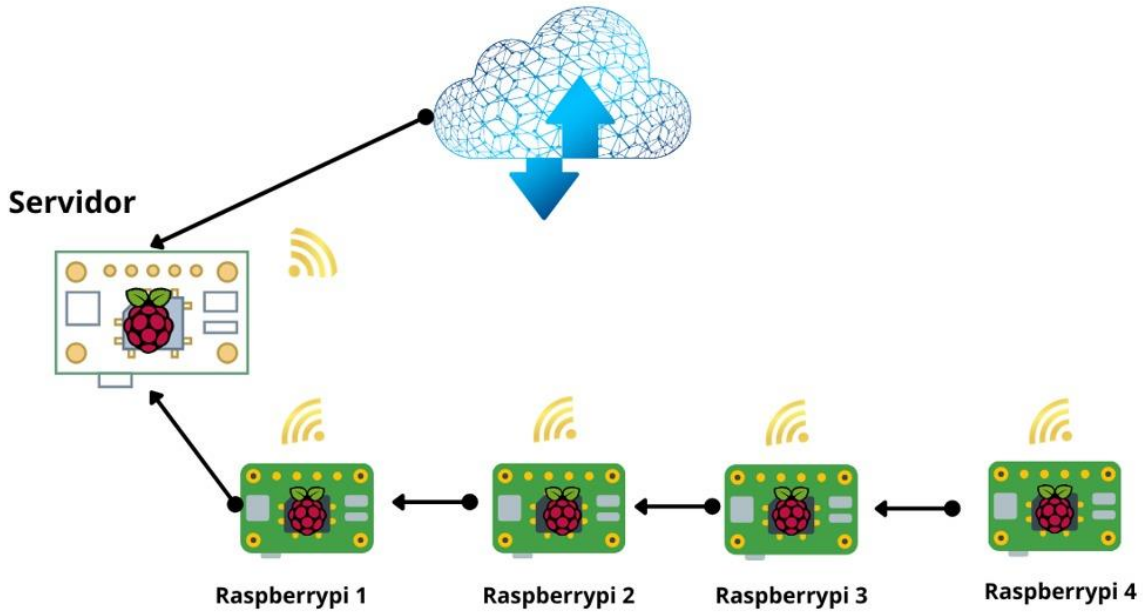


Fig. 42 Estructura final de nodos.

5. Análisis de Resultados

En primer lugar, la conexión entre el dispositivo Raspberry Pi y el módulo RFID se estableció sin problemas, lo que permitió una lectura confiable de los identificadores de tarjetas. Para la lectura de la tarjeta RFID, así como para el envío de los datos a los demás raspberry, se utilizó una arquitectura cliente - servidor haciendo uso de los sockets que permitieron la comunicación entre procesos que se ejecutaron en el mismo host o en diferentes hosts a través de una red.

Lo segundo fue crear una API que nos permita tener el identificador de la tarjeta, por lo que fue necesario crear un servidor con el lenguaje de programación python que se encargó de la lectura y almacenamiento del identificar (ID) usando el framework de flask para la creación de APIs. Luego, en el aplicativo web, necesitamos llamar a la API por lo que creamos un archivo capaz de realizar peticiones al servidor en donde está la API y así mostramos el identificador de la tarjeta en el input del código HTML encargado de recibir esa información. En la (**Fig. 43**) se aprecia la ejecución de la API en un navegador web.

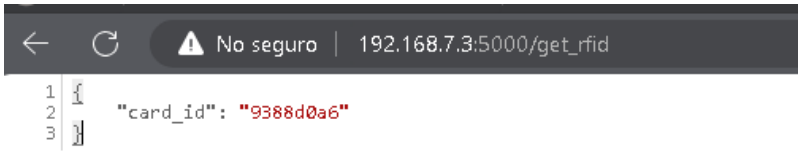


Fig. 43 API del identificador único del estudiante.

A continuación, se procede a mostrar la ejecución del aplicativo web encargado del registro del nuevo estudiante y actualización de este. Primero contamos con el método guardar, este método será de encargado de generar una nueva transacción a la cartera de Metamask y este a la vez será el encargado de almacenar los datos de la transacción en la red de Etherium creando hash que son imposibles de alterar, además de cada transacción se genera un nuevo bloque que a su vez cambia el hash por uno nuevo lo que lo convierte en un bloque inmutable al estar cambiando constantemente su hash.

1. Agregar un nuevo estudiante.

En la (**Fig. 44**) se muestra la implementación del formulario que permite agregar un nuevo estudiante, cuando presione el botón guardar este irá al método de creación en el contrato y mostrará la interfaz de Metamask para poder realizar las transacciones (**Fig. 45**)

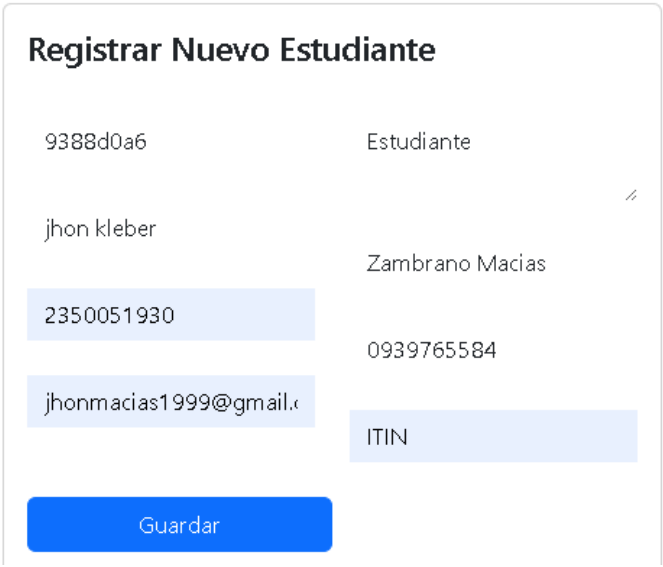


Fig. 44 Formulario para agregar un nuevo estudiante.

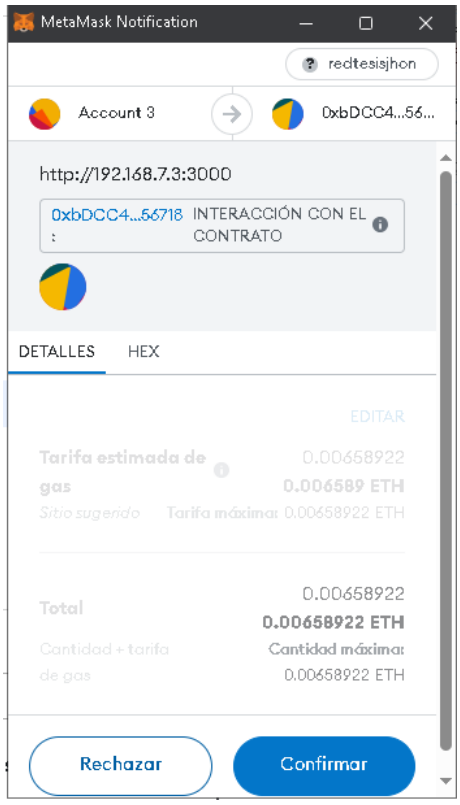


Fig. 45 Confirmar la transacción.

Una vez que se realiza la transacción se muestra en la tabla el nuevo dato creado, cabe recalcar que para listar los datos no se realiza una transacción ya que solo estamos obteniendo los datos (**Fig. 46**).

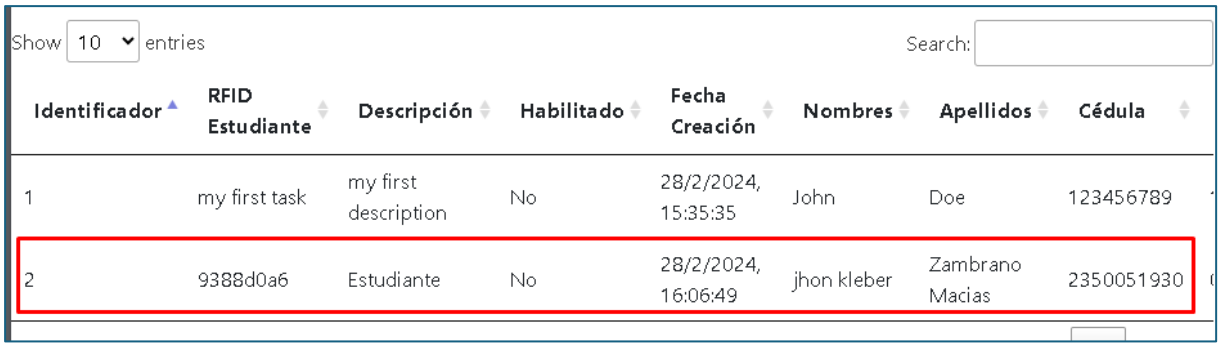


Fig. 46 Listado de estudiantes.

En las aplicaciones descentralizadas la actualización es un poco diferente, ya que actualizar una transacción no es posible. Sin embargo, seguimos una lógica similar a la actualización, lo que hacemos es editar los datos más no la transacción, los datos se actualizan, pero la transacción es una nueva por lo que si se actualiza una transacción se tendría que saber el hash del bloque anterior y prácticamente esto es imposible por la blockchain.

Entonces, cuando se da clic en el botón de editar de la tabla, se cargan en el formulario (**Fig. 47**) los datos que le corresponden al estudiante seleccionado. Luego se realiza la modificación, donde los campos que se pueden editar son todos a excepción del identificador. Cuando se da clic en actualizar busca la función editar creada en el contrato y automáticamente arroja la nueva transacción que se va a realizar (**Fig. 48**). Cuando se acepte la transacción, automáticamente se recarga la página y se puede ver el nuevo dato actualizado, mire la (**Fig. 49**).

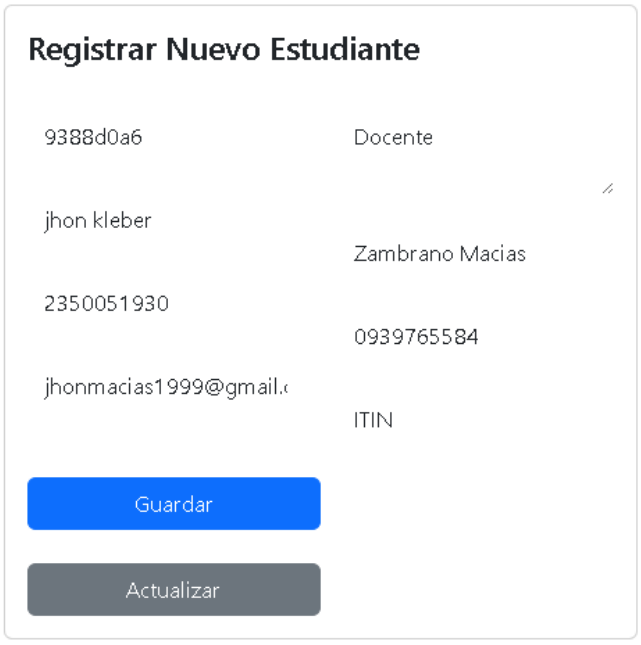


Fig. 47 Formulario para editar un estudiante

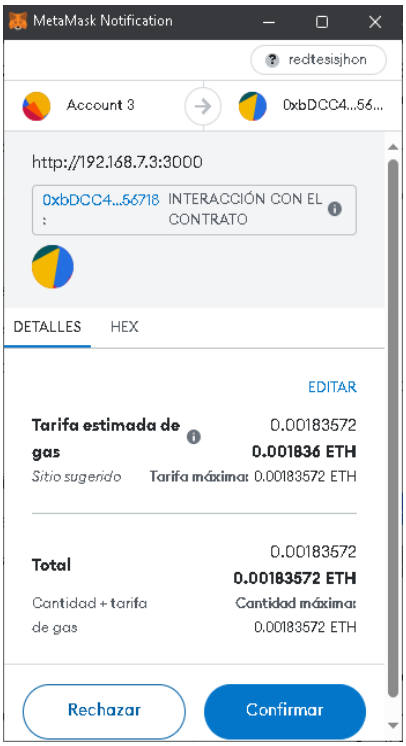


Fig. 48 Transacción para actualizar.

Show 10 entries Search:

Identificador	RFID Estudiante	Descripción	Habilitado	Fecha Creación	Nombres	Apellidos	Cédula
1	my first task	my first description	No	28/2/2024, 15:35:35	John	Doe	123456789
2	9388d0a6	Docente	No	28/2/2024, 16:06:49	jhon kleber	Zambrano Macias	2350051930

Showing 1 to 2 of 2 entries Previous 1 Next

Fig. 49 Lista del estudiante actualizado.

Por último, la infraestructura final (**Fig. 50**) en donde se tiene el módulo RFID que lee las tarjetas y posteriormente se envía al servidor que se ejecuta con Ethereum y la cartera

Metamask, los nodos clientes verifican la información del servidor y se muestran los datos de los estudiantes mediante el aplicativo web.

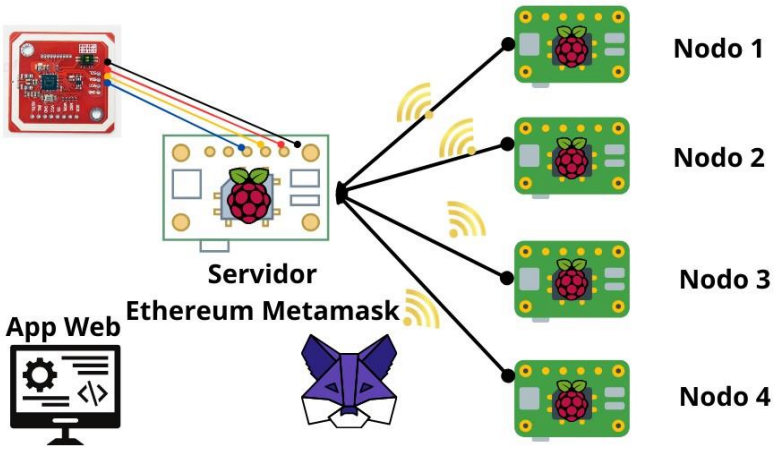


Fig. 50 Infraestructura final.

Finalmente se añadieron todos los nodos (clientes) a la red principal de la blockchain, ayudando a fortalecer y mejorar la red en términos de seguridad, resistencia, descentralización y rendimiento. Es importante mencionar que nuestro servidor principal es aquel que va a ejecutar el nodo principal (**Fig. 51**) y los demás dispositivos se añadirán a ese nodo de la red (**Fig. 52**). Por último, podemos constatar esos nodos añadidos a nuestra red mire la (**Fig. 53**).

```
root@rp4:/home/tesis# geth --http --http.addr "192.168.1.238" --http.port 8545 --syncmode "full" --http.api "eth,net,web3" --http.corsd
omain "*"
INFO [02-29|20:50:33.224] Starting Geth on Ethereum mainnet...
INFO [02-29|20:50:33.226] Bumping default cache on mainnet         provided=1024 updated=4096
INFO [02-29|20:50:33.233] Maximum peer count                       ETH=50 LES=0 total=50
INFO [02-29|20:50:33.237] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or directory"
WARN [02-29|20:50:33.245] Sanitizing cache to Go's GC limits       provided=4096 updated=1263
INFO [02-29|20:50:33.246] Set global gas cap                       cap=50,000,000
INFO [02-29|20:50:33.251] Allocated trie memory caches             clean=189.00MiB dirty=315.00MiB
INFO [02-29|20:50:33.252] Using leveldb as the backing database
INFO [02-29|20:50:33.252] Allocated cache and file handles         database=/root/.ethereum/geth/chaindata cache=629.00MiB handles=524,
288
INFO [02-29|20:50:33.301] Using LevelDB as the backing database
INFO [02-29|20:50:33.304] Opened ancient database                  database=/root/.ethereum/geth/chaindata/ancient/chain readonly=false
INFO [02-29|20:50:33.315] Disk storage enabled for ethash caches   dir=/root/.ethereum/geth/ethash count=3
INFO [02-29|20:50:33.315] Disk storage enabled for ethash DAGs     dir=/root/.ethash              count=2
INFO [02-29|20:50:33.316] Initialising Ethereum protocol           network=1 dbversion=8
INFO [02-29|20:50:33.373]
INFO [02-29|20:50:33.373] ----------------------------------------------------------------------------------------------------------
----------------------------------------
INFO [02-29|20:50:33.373] Chain ID:  1 (mainnet)
INFO [02-29|20:50:33.373] Consensus: Beacon (proof-of-stake), merged from Ethash (proof-of-work)
INFO [02-29|20:50:33.373]
INFO [02-29|20:50:33.373] Pre-Merge hard forks (block based):
INFO [02-29|20:50:33.373]  - Homestead:                   #1150000  (https://github.com/ethereum/execution-specs/blob/master/network-up
grades/mainnet-upgrades/homestead.md)
INFO [02-29|20:50:33.373]  - DAO Fork:                    #1920000  (https://github.com/ethereum/execution-specs/blob/master/network-up
grades/mainnet-upgrades/dao-fork.md)
INFO [02-29|20:50:33.373]  - Tangerine Whistle (EIP 150): #2463000  (https://github.com/ethereum/execution-specs/blob/master/network-up
grades/mainnet-upgrades/tangerine-whistle.md)
INFO [02-29|20:50:33.373]  - Spurious Dragon/1 (EIP 155): #2675000  (https://github.com/ethereum/execution-specs/blob/master/network-up
grades/mainnet-upgrades/spurious-dragon.md)
INFO [02-29|20:50:33.373]  - Spurious Dragon/2 (EIP 158): #2675000  (https://github.com/ethereum/execution-specs/blob/master/network-up
grades/mainnet-upgrades/spurious-dragon.md)
INFO [02-29|20:50:33.373]  - Byzantium:                   #4370000  (https://github.com/ethereum/execution-specs/blob/master/network-up
grades/mainnet-upgrades/byzantium.md)
INFO [02-29|20:50:33.373]  - Constantinople:              #7280000  (https://github.com/ethereum/execution-specs/blob/master/network-up
grades/mainnet-upgrades/constantinople.md)
```

Fig. 51 Ejecución del nodo principal

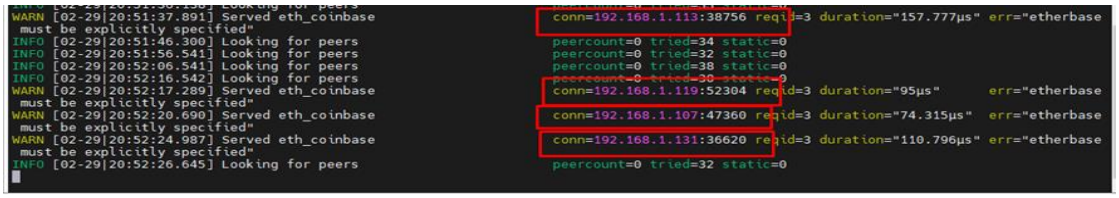


Fig. 52 Dispositivos añadidos al nodo principal.

Fig. 53 Nodos añadidos.

6. Conclusiones y recomendaciones

Con esta propuesta de trabajo de integración de tecnologías IoT y Blockchain, se obtuvo un sistema completo que demuestra ser una solución bastante novedosa y potencial para hacer más seguros y eficientes los sistemas de control de acceso a instituciones educativas. Al unir la capacidad de los dispositivos IoT, como la Raspberry Pi con la tecnología Blockchain, se consiguió crear un ambiente seguro y confiable para registrar y verificar el ingreso de estudiantes a la Universidad de las Fuerzas Armadas ESPE.

La configuración del módulo RFID representa un componente crítico para el éxito del sistema de control de ingreso de estudiantes, es esencial para garantizar una integración eficiente en conjunto con los dispositivos IoT y la tecnología Blockchain. La elección adecuada de los tags y lectores RID contribuyen directamente en la eficacia en términos de identificación y detección de usuarios autorizados permitiendo así una construcción adecuada de registros inmutables en la Blockchain.

Al incorporar la tecnología Blockchain en el sistema de control de ingreso de estudiantes, se aseguró la integridad de los registros y la autenticación de los usuarios. La utilización de cadenas de bloques inalterables elimina la posibilidad de manipulación de la información, este es un aspecto importante en entornos críticos y aplicaciones que demandan un alto nivel de seguridad.

El sistema permite recopilar el id en tiempo real sobre las interacciones de los usuarios, lo que simplifica el monitoreo y facilita la toma de decisiones basadas en información actualizada. Esta capacidad resulta particularmente crucial en situaciones que requieren un estricto control de acceso, como en edificios gubernamentales, instalaciones militares y empresas de alto valor.

La utilización de contratos inteligentes en un sistema de control de ingreso basado en IoT y Blockchain aporta una eficiencia y precisión mejoradas en la gestión de permisos y registros de ingreso. Al programar reglas y condiciones específicas en estos contratos, se establece un mecanismo automatizado de control que verifica y autoriza los accesos de manera rápida y confiable.

De acuerdo con la hipótesis que se planteó, concluimos que es factible la integración de las tecnologías Blockchain e IoT en la optimización de gestión de ingreso de estudiantes. Esto se debe a que cumple con las medidas de seguridad, eficiencia y transparencia en el manejo de los datos de los estudiantes, logrando con esto mejorar el ingreso de los estudiantes en la Universidad de las Fuerzas Armadas ESPE.

Recomendaciones

Es importante llevar a cabo pruebas exhaustivas de seguridad para detectar posibles vulnerabilidades y riesgos en el sistema. Este paso crítico posibilita la adopción de medidas necesarias para reforzar la protección de la información sensible, prevenir ataques o intrusiones y garantizar la robustez del sistema frente a posibles amenazas.

Es necesario realizar un diseño y desarrollo minucioso de contratos inteligentes para asegurar su fiabilidad y seguridad. Esto requiere seguir las mejores prácticas de programación, como la adopción de bibliotecas especializadas en seguridad, llevar a cabo pruebas de seguridad y realizar auditorías. Es crucial gestionar de manera adecuada los fondos vinculados a cada contrato, evitando potenciales vulnerabilidades y protegiéndolos contra posibles ataques.

Es esencial que cada estudiante cuente con su propio tag o tarjeta RFID para facilitar la identificación al ingresar a la Universidad de las Fuerzas Armadas ESPE Sede Santo Domingo. Esta medida contribuirá al mantenimiento del orden en los registros de ingreso y fortalecerá la seguridad de los datos frente a posibles incidentes.

Se recomienda a la Universidad de las Fuerzas Armadas ESPE Sede Santo que, para la implementación del sistema, asignen a cada estudiante una tarjeta o tag RFID de alta frecuencia (HF) con dimensiones de 85.6 mm x 54 mm y fabricada en material plástico. Esta tarjeta se utilizará para controlar su ingreso. El lector RFID se podrá instalar en un punto de control de ingreso común.

El sistema es escalable en infraestructura; por lo tanto, se sugiere no solo implementarlo como sistema de control de ingreso para estudiantes, sino también para el personal administrativo y militar. Para la implementación total del sistema es requerido que cada nodo se ubique tomando en consideración la ubicación y cobertura de la señal inalámbrica.

La elección de la plataforma Blockchain es crucial y existen diversas opciones, como Ethereum, Fabric o Corda. Es esencial llevar a cabo un análisis detallado de las características y capacidades de cada plataforma para seleccionar la más idónea de acuerdo con los requisitos del sistema. Factores como la escalabilidad, seguridad, capacidad para ejecutar contratos inteligentes y la existencia de una comunidad de desarrolladores activa deben ser cuidadosamente considerados durante el proceso de selección.

7. Referencias bibliográficas

[1] G. Rekha and B. U. Maheswari, "Raspberry Pi Forensic Investigation and Evidence Preservation using Blockchain," 2021 International Conference on Forensics, Analytics, Big Data, Security (FABS), Bengaluru, India, 2021, pp. 1-5, doi: 10.1109/FABS52071.2021.9702622.

[2] V. Román and J. Ordieres-Meré, "[WiP] IoT Blockchain Technologies for Smart Sensors Based on Raspberry Pi," 2018 IEEE 11th Conference on Service-Oriented Computing and Applications (SOCA), Paris, France, 2018, pp. 216-220, doi: 10.1109/SOCA.2018.00038.

[3] A. P. Junfithrana, E. Liani, M. Z. Suwono, D. Meldiana and A. Suryana, "Rice Donation System in Orphanage Based on Internet of Things, Raspberry-Pi, and Blockchain," 2018 International Conference on Computing, Engineering, and Design (ICCED), Bangkok, Thailand, 2018, pp. 235-238, doi: 10.1109/ICCED.2018.00053.

[4] A. Kaushik and A. S. Pillai, "Blockchain and IoT based inventory monitoring system," 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2021, pp. 827-831, doi: 10.1109/ICESC51422.2021.9532876.

[5] A. Mohammed, H. Nahom, A. Tewodros, Y. Habtamu and G. Hayelom, "Deep Reinforcement Learning for Computation Offloading and Resource Allocation in Blockchain-Based Multi-UAV-Enabled Mobile Edge Computing," 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 2020, pp. 295-299, doi: 10.1109/ICCWAMTIP51612.2020.9317445.

[6] B. N. Gangothri, K. P. Satamraju and B. Malarkodi, "Sensor-based Ambient Healthcare Architecture using Blockchain and Internet of Things," 2023 10th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 2023, pp. 543-546, doi: 10.1109/SPIN57001.2023.10116867.

[7] Z. Gong-Guo and Z. Wan, "Blockchain-based IoT security authentication system," 2021 International Conference on Computer, Blockchain and Financial Development (CBFD), Nanjing, China, 2021, pp. 415-418, doi: 10.1109/CBFD52659.2021.00090.

[8] A. Dharani and S. M. Khaliq-ur-Rehman Raazi, "Integrating Blockchain with IoT for Mitigating Cyber Threat in Corporate Environment," 2022 Mohammad Ali Jinnah

University International Conference on Computing (MAJICC), Karachi, Pakistan, 2022, pp. 1-6, doi: 10.1109/MAJICC56935.2022.9994206.

[9] A. Mohammed, H. Nahom, A. Tewodros, Y. Habtamu and G. Hayelom, "Deep Reinforcement Learning for Computation Offloading and Resource Allocation in Blockchain-Based Multi-UAV-Enabled Mobile Edge Computing," 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 2020, pp. 295-299, doi: 10.1109/ICCWAMTIP51612.2020.9317445

[10] Dominique Paret; Jean-Paul Huon, "Overall Architecture of the IoT Chain," in Secure Connected Objects, Wiley, 2017, pp.87-87, doi: 10.1002/9781119426639.part3.

[11] Guaña-Moya et al, "Tecnología Blockchain, qué es y cómo funciona," Revista Ibérica De Sistemas e Tecnologias De Informação, pp. 101-114, 2022. [En línea]. Disponible en: https://ezp1.espe.edu.ec:9443/login?url=https://www.proquest.com/scholarly-journals/tecnología-blockchain-qué-es-y-cómo-funciona/docview/2812105701/se-2.

[12] I. S. Moreno “Introducción al blockchain y criptomonedas en 100 preguntas”, pp. 30-40, 2021. [En línea]. Disponible en ttps://books.google.es/books?hl=es&lr=&id=nqxvEAAAQBAJ&oi=fnd&pg=PT9&dq=tipos+de+blockchain&ots=txWD7u1jW8&sig=5gpetwbEsSvg6HRD8nfMK3EdFEs#v=onepage&q=tipos%20de%20blockchain&f=false

[13] S. Amsler and S. Shea, “RFID o identificación por radiofrecuencia,” ComputerWeekly.es, Apr. 14, 2021 [En línea]. Disponible en: https://www.computerweekly.com/es/definicion/RFID-o-identificacion-por-radiofrecuencia

[14] J. P. Cevallos, “¿Qué es Ethereum? | ethereum.org,” ethereum.org, 2019 [En línea]. Disponible en: https://ethereum.org/es/what-is-ethereum/

[15] Chainlink, “What Are Smart Contracts in Blockchain? “, 2023 [En línea]. Disponible en: https://chain.link/education/smart-contracts

[16] Simplilearn, “What is Solidity Programming: Data Types, Smart Contracts, and EVM”, 2023 [En línea]. Disponible en: https://www.simplilearn.com/tutorials/blockchain-tutorial/what-is-solidity-programming

[17] S. C. Team, “What is Truffle in Blockchain? – Explained”, Shardeum Blogs | EVM-based Sharded L1 Blockchain, 2024, [En línea]. Disponible en: https://shardeum.org/blog/truffle/#:~:text=Truffle%20is%20a%20development%20framework,testing%2C%20deployment%2C%20and%20management.

[18] M. H. D. Phillips, “What is MetaMask? How to Use the Top Ethereum Wallet”, Decrypt, 2022. [En línea]. Disponible en: https://decrypt.co/resources/metamask

[19] G. Weston, “What is Ganache Blockchain”, 101 Blockchains, 2023. [En línea] Disponible en: https://101blockchains.com/ganache-blockchain/

[20] J. Zambrano, L. Olalla, “Solidity-Javascript-RaspberryPI”, 2023[En línea]. Disponible en: https://github.com/JhonZ-Dev/Solidity-Javascript-RaspberryPI.git