UNIVERSIDAD AUTÓNOMA DEL
ESTADO DE QUINTANA ROO

DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

Integración de la Tecnología Blockchain en un Sistema de Posicionamiento en Espacios Interiores

TESIS

PARA OBTENER EL GRADO DE

INGENIERÍA EN REDES

PRESENTA

DYEL ALDEMIR ACOSTA

DIRECTOR DE TESIS

Dr. HOMERO TORAL CRUZ

ASESORES

Dr. Julio César Ramírez Pacheco

MM. Jesús Orifiel Álvarez Ruiz

Dr. Ismael Osuna Galán

Dr. José Antonio León Borges

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO
ÁREA DE TITULACIÓN

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO
DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

CHETUMAL, QUINTANA ROO, MÉXICO, NOVIEMBRE DE 2023

UNIVERSIDAD AUTÓNOMA DEL
ESTADO DE QUINTANA ROO

DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

Tesis elaborada bajo la supervisión del Comité de Tesis del programa de licenciatura y aprobada como requisito para obtener el grado de:

INGENIERÍA EN REDES

COMITÉ DE TESIS

Director: Dr. Homero Toral Cruz

Asesor: Dr. Julio César Ramírez Pacheco

Asesor: MM. Jesús Orifiel Álvarez Ruiz

Asesor: Dr. Ismael Osuna Galán

Asesor: Dr. José Antonio León Borges

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO
ÁREA DE TITULACIÓN

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO
DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

CHETUMAL, QUINTANA ROO, MÉXICO, NOVIEMBRE DE 2023

Agradecimientos

Me gustaría expresar mi más profundo agradecimiento a las personas que más me han ayudado a lo largo de mi carrera. En primer lugar, deseo expresar mi más profundo agradecimiento a mis padres, Elena Acosta y Jose Acosta, así como a mis hermanos, Shakir Acosta y Herbert Acosta. Su constante respaldo, aliento y comprensión han sido la fuente de la fortaleza necesaria para superar los desafíos que he enfrentado a lo largo de mi vida.

También, quisiera expresar mi agradecimiento a mi director de tesis, Dr. Homero Toral, por haberme apoyado constantemente durante toda mi investigación con su valiosa conocimiento, amabilidad y paciencia. Me brindó la libertad de trabajar a mi manera. Además, me gustaría extender mi agradecimiento a mis profesores que me transmitieron sus conocimientos. En conjunto, han contribuido significativamente a mi crecimiento como ingeniero y no puedo expresar del todo lo genuinamente agradecido que estoy por su orientación y tutoría.

A mis amigos por la oportunidad de colaborar con ellos en varios proyectos de investigación a lo largo de mi carrera de ingeniería cual ha sido un inmenso privilegio, marcado por numerosos momentos encantadores que siempre apreciaré. Quiero expresar un reconocimiento particular a mis amigos más cercanos, Leider, Kevin, Michael y Leo-Eli por su apoyo incondicional. Su presencia ocupa un espacio único y duradero en mi corazón, y soy optimista de que nuestro vínculo perdurará la prueba del tiempo, fomentando una amistad que durará toda la vida.

RESUMEN

Esta de tesis presenta la integración de la tecnología blockchain en un sistema de posicionamiento en espacios interiores (IPS). Para estos sistemas, la integración de la seguridad mediante servidores descentralizados y distribuidos es importante ya que diversas industrias, como la médica, han comenzado a utilizar dispositivos basados en sensores de baja potencia. La inclinación de estos dispositivos hacia el bajo consumo los hace propensos a riesgos de seguridad e intrusiones cibernéticas. Blockchain asciende como una solución progresiva, evidenciando características distintivas como mecanismos de contabilidad descentralizados, algoritmos de consenso, funcionalidad dinámica de contratos inteligentes, criptografía y funciones hash. Por lo tanto, blockchain se convierte en una tecnología adecuada para proteger sistemas IPS e incluso dispositivos IoT de baja potencia.

La implementación del sistema se realizó utilizando la plataforma BigchainDB, la cual ofrece alto rendimiento, baja latencia, control descentralizado y almacenamiento de datos inmutable; construido como una base de datos con características blockchain. El sistema propuesto está compuesto por una computadora de placa única Raspberry Pi, sensores iBeacons conectados vía Bluetooth, un servidor local con sistema operativo Ubuntu 18.04.6 LTS que alberga la base de datos MariaDB y la blockchain BigchainDB.

En el desarrollo de la gestión de datos se crearon códigos para el uso de “MongoDB”, reconocido por su escalabilidad y alto rendimiento, siendo ideal para datos estructurados y no estructurados con potencial de rápido crecimiento.

Palabras Clave: Blockchain, BigchainDB, iBeacons, Raspberry pi, MongoDB, IPS.

ABSTRACT

This thesis presents the integration of Blockchain technology in an indoor positioning system (IPS). For these systems, the integration of security through decentralized and distributed servers is important because some industries, such as medical, have started to use low-powered sensor-based devices. The inclination of these devices towards low power makes them prone to security risks and cyber intrusions. Blockchain ascends as a progressive solution, evidencing distinguishing features such as decentralized ledger mechanisms, consensus algorithms, dynamic smart contracts functionality, cryptography and hash functions. Thus, blockchain becomes a suitable technology for securing IPS systems and even IoT low-powered devices.

The implementation of the system was carried out using the BigchainDB platform, which offers high throughput, low latency, decentralized control, and immutable data storage; built like a database with blockchain characteristics. The proposed system is composed of a Raspberry Pi single board computer, iBeacons sensors connected via Bluetooth, a local server with Ubuntu 18.04.6 LTS operating system that houses the MariaDB database, and the BigchainDB blockchain.

In the development of data management, codes were created for the use of "MongoDB", recognized for its scalability and high performance, being ideal for structured and unstructured data with the potential for rapid growth.

Key words: Blockchain, BigchainDB, iBeacons, Raspberry pi, MongoDB, IPS**.**

Tabla de Contenido

Lista de Tablas

Lista de Figuras

INTRODUCCIÓN

A medida que la demanda de comunicaciones inalámbricas continúa aumentando, existe una presión creciente sobre los recursos de detección de espectro (Panarello, Tapas, Merlino, Longo, & Puliafito, 2018) (Caramés & Lamass, 2018), mientras que la tasa de utilización de la detección de espectro disponible sigue siendo relativamente baja (Koyuncu & Yang, 2010) (Mahtab, 2009) (Bardwell & Akin, 2005). En respuesta a este desafío, las redes inalámbricas cognitivas ofrecen una solución al permitir el acceso dinámico del espectro a las bandas de frecuencia autorizadas. Este enfoque tiene como objetivo maximizar la utilización de las brechas de espectro, mejorar la utilización general del espectro y aliviar la escasez de recursos de espectro. Muchas aplicaciones del mercado masivo requieren capacidades de posicionamiento exactas en todo tipo de entornos. Aunque los sistemas de posicionamiento satelital proporcionan un rendimiento excelente del posicionamiento en exteriores, no se puede decir los mismo para el posicionamiento en interiores (Mautz, 2012) (Seunghyeon, Seok, Lee, & In, 2022) (Boneh & Shoup, 2023). Por esta razón se hizo evidente que el posicionamiento en interiores se ha convertido en un foco de investigación y desarrollo durante la década pasada. Se requiere un sistema con una técnica de localización, tecnologías de localización y, además, una red de sensores para localizar y rastrear objetos dentro de edificios y ambientes cerrados (Koyuncu & Yang, 2010). Las coordenadas de un plano se debe estimar por el sistema. Las diferentes técnicas y soluciones se basan en métodos de triangulación y multicapa mediante señales de luz, ultrasonido o radio, que devuelven información posicional. Estos métodos tienen diferentes arquitecturas para determinar la ubicación de los objetos, elegir el método adecuado para el entorno adecuado hará que los resultados sean menos propensos a errores.

Uno de los paradigmas revolucionarios que ha introducido nuevos conceptos en el intercambio seguro de datos e información es blockchain. Inicialmente estaba destinado a la conocida criptomoneda Bitcoin. Sin embargo, a lo largo de los últimos años, se ha ido implementando en diversos campos: Sistema de Posicionamiento Global (GPS), sanidad (asuntos médicos), Internet de las Cosas (IOT), y transporte inteligente. Si se usa en GPS, puede crear un sistema de detección de espectro confiable (Mautz, 2012). La tecnología blockchain se puede usar para mejorar la eficiencia energética y la precisión de la detección de espectro en redes inalámbricas cognitivas. Esta técnica segura de detección de espectro permite la detección colaborativa, la adaptabilidad a los cambios ambientales y el ajuste en tiempo real de los nodos participantes. Incorpora un algoritmo de evaluación para evaluar la fiabilidad y confiabilidad de los nodos de

detección, calculando sus valores de confianza. El sistema registra el consumo de energía y el rendimiento de detección de cada nodo y almacena los valores de confianza en una lista de confiabilidad cifrada administrada por la autoridad central de la cadena de bloques. Al considerar tanto la eficiencia energética como la precisión de detección, el algoritmo propuesto extiende la vida útil operativa de las redes inalámbricas cognitivas, según lo respaldan los datos experimentales en (Koyuncu & Yang, 2010).

Capítulo I

1. PLANTEAMIENTO DEL PROBLEMA

1.1 PROBLEMÁTICA

Los sistemas de posicionamiento global (GPS) son altamente efectivos en entornos exteriores, pero su funcionamiento se ve obstaculizado en entornos interiores como edificios, almacenes, fabricas, hospitales o centros comerciales. La creciente demanda de servicios basados en la localización en el ámbito de los espacios interiores causa la necesidad de localizar objetos con precisión, y es crucial para optimizar la eficiencia de las operaciones y garantizar la seguridad.

Actualmente, estudios sobre métodos de posicionamiento en interiores se centran en la precisión del posicionamiento, mientras que el debate sobre consideraciones de seguridad y privacidad es limitado. Los sistemas existentes de posicionamiento utilizan tecnologías como GPS, Wi-Fi o Bluetooth, pero estas soluciones presentan limitaciones en términos de precisión y seguridad. Estas tecnologías se ven obstaculizadas por problemas como el desfase en la ubicación de los objetos, la interferencia de señales o la falta de protección contra ataques cibernéticos.

En el presente trabajo de tesis se realizará la aplicación de la tecnología blockchain en un sistema de posicionamiento de objetos en espacios interiores en una red de sensores inalámbricos.

1.2 OBJETIVOS

1.2.1 Objetivo General

Aplicar la tecnología blockchain en un sistema de posicionamiento de objetos en espacios interiores.

1.2.2 Objetivo Específicos

- Recolectar en la estación base (Raspberry PI 3 B+) los valores RSSI (Received Signal Strength Indicator) de los nodos de la red de sensores inalámbricos.
- Procesar los valores RSSI en la estación base, mediante un script desarrollado en Python.
- Organizar la información de cada nodo y enviar a la nube mediante solicitud HTTP por el método GET; para su almacenamiento en un servidor local.
- Registrar la información recibida en el servidor local, paralelamente en un servidor descentralizado y distribuido, que utiliza la tecnología blockchain.

1.3 ALCANCES Y LIMITACIONES

1.3.1 Alcance

El alcance de esta investigación abarca un examen del entorno tecnológico de blockchain, con un enfoque específico en explorar posibles aplicaciones dentro del ámbito de sistemas que utilizan dispositivos de sensores de baja potencia. El objetivo principal es comparar las distintas plataformas de blockchain para evaluar la viabilidad de integrar la tecnología de blockchain y mejorar la seguridad de los sistemas de posicionamiento empleados en espacios interiores. De esta manera, la diferenciación entre las diversas plataformas de blockchain facilitará la conceptualización de las ventajas inherentes que se pueden aprovechar mediante la utilización de esta tecnología.

1.3.2 Limitaciones

A pesar de sus numerosos beneficios, la tecnología blockchain no está exenta de limitaciones y desafíos. Este documento examina las ventajas de seguridad que blockchain puede ofrecer a los sistemas de posicionamiento en interiores (IPS) y evalúa su relevancia para los sistemas de Internet de las cosas (IoT). Aunque el enfoque principal es la implementación de blockchain, es esencial reconocer que otras características también merecen consideración. Esta es una limitación que queda pendiente para futuras investigaciones. Dentro del alcance de este trabajo, la implementación de blockchain se realiza en un modelo específico de Raspberry Pi y una versión específica de Ubuntu. El estudio de caso se limita a una única versión de Ubuntu (18.04.6). Además, el trabajo se centra en un sistema centralizado para medir la transacción de

datos destinados para asegurar los mismos datos del IPS, constituyendo otra limitación. Otras limitaciones generales adicionales son las siguientes:

- Escalabilidad
- Latencia
- Rendimiento
- Tamaño y ancho de banda
- Brechas de seguridad
- Escalabilidad

1.4 VIABILIDAD

Teniendo en cuenta los diferentes aspectos discutidos de la tecnología blockchain, el proyecto parece viable. La exploración centrada en dispositivos sensores de baja potencia como los sensores iBeacons y la evaluación de plataformas blockchain para mejorar los sistemas de posicionamiento en interiores se alinean con las tendencias tecnológicas actuales. El reconocimiento de las limitaciones, incluidas las relacionadas con el hardware específico y el sistema centralizado, demuestra un enfoque realista. Sin embargo, el éxito del proyecto dependería de abordar estas limitaciones y adoptar soluciones para garantizar la escalabilidad y descentralización.

1.5 JUSTIFICACIÓN

En la actualidad, para ubicar un objeto en entornos interiores no se ha definido un método que sea preciso y que permita la identificaron exacto de tal objeto. Es la razón que el tema de posicionamiento en interiores se ha convertido en un desafío. Los sistemas de posicionamiento en interiores, conocidos como IPS, son utilizados para ubicar objetos dentro de ambientes cerrados. Sin embargo, estos sistemas pueden presentar vulnerabilidades en cuanto a la seguridad de la información transmitida por los sensores y el medio que se transfieren los datos. Existen varios factores que debilitan una transmisión inalámbrica; tales como, atenuación y atenuación por distorsión, perdida en el espacio libre, ruido térmico, absorción atmosférica, mutlicaminos, y refacción. Por tanto, es necesario implementar medidas de autenticación, protección de datos y resistencia a ataques para garantizar la integridad y confiabilidad de los datos obtenidos.

Dado que el Sistema de Posicionamiento Global (GPS) es una tecnología ampliamente utilizada para determinar la ubicación precisa de objetos en espacios exteriores, se deben considerar otros enfoques para localizar objetos en otros tipos de escenarios.

En el presente proyecto se propone aplicar la tecnología blockchain en un sistema de posicionamiento de objetos en espacios interiores en una red de sensores inalámbricos.

Capítulo II

2. MARCO TEÓRICO

2.1 DEFINICIÓN

Blockchain, como sugiere su nombre, se puede visualizar como una cadena de bloques unidos entre sí. En los bloques puede haber cualquier tipo de datos, desde texto, imágenes, hasta audio y video. Pero lo que hace blockchain diferente y único, es su naturaleza inmutable y distribuida. Hay muchos nodos en una red blockchain y cada nodo tiene la misma copia de los datos conocidos como libro mayor. Un libro mayor tiene la propiedad única de ser inmutable, una vez que se escrito en el libro mayor, estaría allí para siempre.

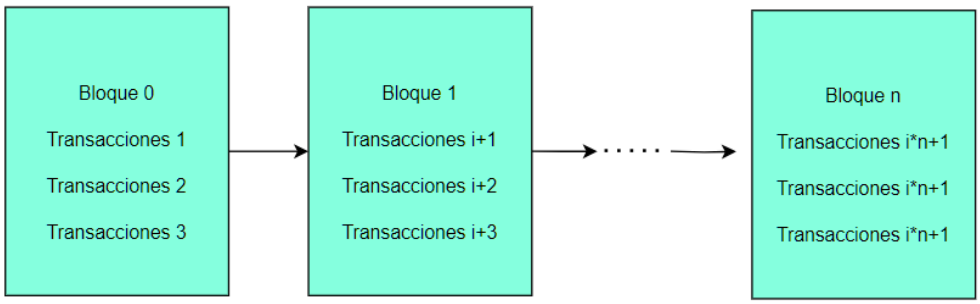

Figura 1 Blockchain: Bloques enlazados en forma de cadena

2.2 HISTORIA

2.2.1 Al principio

La historia de la tecnología blockchain tiene sus raíces a mediados de la década de 1990 cuando dos científicos, Stuart Haber y W Scott Stornetta, buscaron soluciones para garantizar la seguridad e integridad de la información digital, evitando alteraciones no autorizadas. Propusieron el uso de una cadena de bloques protegidos criptográficamente para lograr este objetivo, y sus ideas se publicaron por primera vez en un artículo en 1991.

A medida que creció la necesidad de contramedidas contra el spam y los abusos organizacionales en 1993, se estableció el concepto de prueba de trabajo para abordar estas preocupaciones.

El momento crucial para blockchain ocurrió en 2008 cuando una figura enigmática conocida como Satoshi Nakamoto presentó un documento técnico seminal titulado "Bitcoin: A Peer-to-Peer E-Cash System". Este documento introdujo el concepto de cadena de bloques en el contexto de una moneda digital descentralizada, Bitcoin. Describía un método seguro para intercambiar fondos, combinando criptografía, ingeniería informática y teoría de juegos. Sin embargo, es esencial tener en cuenta que, si bien Bitcoin es un ejemplo ampliamente reconocido de tecnología de cadena de bloques, los términos "Bitcoin" y "cadena de bloques" no son sinónimos (Carpena, 2018).

En 2014, Ethereum surgió como un avance significativo en la tecnología blockchain. A diferencia de Bitcoin, Ethereum va más allá de ser una mera moneda digital o un libro de transacciones. Introdujo la máquina virtual Ethereum, a menudo denominada "computadora mundial", que permite la ejecución de contratos inteligentes. Estos contratos inteligentes se pueden implementar en la cadena de bloques, lo que permite a los desarrolladores crear aplicaciones descentralizadas (dApps) que se distribuyen y operan en la propia cadena de bloques.

2.2.2 Años Significativos

En ***2015***, el Bitcoin (conocida como la criptomoneda pionera basada en la tecnología de cadena de bloques) comenzó a generar interés en las personas, ya se estaba poniendo atractivo. Consecuentemente, la gente comenzó a tomarlo más en serio como un archivo digital viable y

una alternativa potencial a las monedas tradicionales. La naturaleza descentralizada y la tecnología blockchain subyacente comenzaron a intrigar no solo a los entusiastas de la tecnología, sino también a los inversores y las instituciones financieras.

En ***2016***, el concepto de blockchain en sí mismo comenzó a recibir una gran atención por parte de varias industrias, particularmente del sector financiero. La naturaleza inmutable y transparente de la tecnología blockchain presentó una solución prometedora para varios desafíos que enfrenta la industria financiera, como reducir el fraude, mejorar la seguridad y mejorar la eficiencia de las transacciones y el mantenimiento de registros.

En ***2017***, el potencial de la tecnología blockchain se hizo más evidente y ganó reconocimiento como una tecnología fundamental con implicaciones de gran alcance. La prestigiosa Harvard Business Review publicó un artículo proponiendo blockchain como tecnología fundamental, enfatizando sus capacidades transformadoras más allá de las criptomonedas como Bitcoin. El artículo destacó las aplicaciones potenciales de blockchain en varias industrias, como la gestión de la cadena de suministro, la atención médica, los bienes raíces y más.

Este reconocimiento por parte de Harvard Business Review solidificó aún más el estado de blockchain como una tecnología revolucionaria con el potencial de interrumpir los sistemas y procesos tradicionales en múltiples sectores. Como resultado, el interés en la tecnología blockchain se disparó, lo que llevó a una mayor investigación, inversión y desarrollo en el campo. Posteriormente, numerosas empresas y organizaciones comenzaron a explorar formas de implementar soluciones de cadena de bloques en sus operaciones, lo que marcó el comienzo de una adopción e integración más amplias de esta tecnología transformadora en el mundo empresarial.

2.2.3 La evolución

Ahora, en el ámbito de la tecnología, la cadena de bloques (blockchain) ha captado la atención de muchos. Su atractivo se ve realzado por un velo de anonimato que envuelve a este prodigio, como aquellos enigmáticos tesoros digitales llamados criptomonedas, con Bitcoin como su emblema más enigmático. Sin embargo, bajo esta apariencia de secreto se encuentra el verdadero encanto de la tecnología: su transparencia sin igual (POPOVSKI & GEORGE SOUSSOU). Así mismo, la aplicación de blockchain llega a lo largo y ancho, entretejiendo sus aplicaciones en casi todas las industrias.

Un ejemplo más fascinante se encuentra en el uso que dio Ethereum, donde surgió una plataforma descentralizada repleta de contratos inteligentes. Estos contratos, imbuidos de capacidades extraordinarias, “permitieron a los desarrolladores crear mercados, registrar deudas y promesas, y conjurar el movimiento de fondos de acuerdo con instrucciones dadas hace mucho tiempo (como un testamento o un contrato de futuros) y muchas otras cosas que aún no se han inventado, todo sin la necesidad de intermediarios o del riesgo de contraparte”, según afirman (POPOVSKI & GEORGE SOUSSOU).

Es con Ethereum que las industrias han comenzado sus primeras exploraciones audaces en territorios desconocidos. A diferencia del propósito singular de Bitcoin, una mera moneda en el reino de las monedas digitales, Ethereum se revela como una herramienta (ledger) superior, una herramienta de posibilidades donde se pueden conjurar y aplicar nuevos programas

2.2.4 El futuro

Se llevó a cabo un taller público cerca del final de cada una de las varias etapas del proceso de selección. La etapa de presentación dio inicio a este proceso, cual se culminó el 15 de mayo de 1998. Mientras tanto, el inicio oficial de la primera ronda de evaluación se llevó a cabo en California, Ventura, del 20 al 22 de agosto de 1998. Se denominó The First Advanced Encryption Standard Candidate Conference y todos los candidatos aceptados debían presentarse aquí y ser juzgados por la comunidad criptográfica internacional.

2.3 VENTAJAS Y DESVENTAJAS

Las siguientes son las ventajas de Blockchain:

- La tecnología blockchain es un sistema descentralizado y es el principal beneficio de esta tecnología.
- El sistema funciona sin intermediarios y todos los participantes de esta blockchain toman las decisiones.
- Cada acción se registra en blockchain y los datos de los registros están disponibles para todos los participantes de esta Blockchain y no se pueden cambiar ni eliminar. Los resultados de esta grabación otorgan transparencia,

inmutabilidad y confianza a blockchain (Krichen, Ammi, Mihoub, & Almutiq, 2022) (Kumar, Khan, Kadry, & Rho, 2021).

- La confianza de Blockchain se basa en la creencia de dos o más participantes, que no se conocen. La idea principal son las transacciones reales y no inútiles entre estas personas desconocidas. La confianza se puede aumentar aún más, porque puede haber más procesos y registros compartidos (Seunghyeon, Seok, Lee, & In, 2022).
- Nadie puede piratear su información personal.

Las siguientes son las desventajas de blockchain:

- La principal desventaja de blockchain es el alto consumo de energía. El consumo de energía es necesario para mantener un libro de contabilidad en tiempo real.
- Puede ser atacado por varios protocolos como PoW y PoS.
- Difícil de implementar con software.

2.4 CONCEPTOS PARA ENTENDER BLOCKCHAIN

Es de gran importancia la comprensión de los conceptos y principios de funcionamiento de blockchain para poder aplicarlos en aplicaciones y tecnologías potenciales, como Internet de las cosas. Esta sección tiene la tarea, no solo de dar una breve introducción, sino también de explicar los conceptos involucrados para lograr la inmutabilidad, seguridad e integridad de los contenidos almacenados de cada bloque (Ali, y otros, 2019) (Salman, Zolanvari, Erbad, Jain, & Samaka, 2019).

Los sistemas de blockchain se clasifican como una amalgama de criptografía, infraestructura de clave pública y modelado económico, aplicados a redes peer-to-peer y consenso descentralizado para lograr la sincronización de bases de datos distribusidas (Ali, y otros, 2019).

Es útil en la forma en que mantiene registros de las transacciones que ocurren dentro de una red, por lo que se denomina "libro mayor distribuido", lo que significa que su cadena de bloques es una estructura de datos distribuidos.

2.4.1 Descentralización

Los sistemas distribuidos son un paradigma informático en el que dos o más nodos trabajan entre sí en una manera coordinada para lograr un resultado común (Bashir, 2018).

En las infraestructuras de red que están centralizadas, la validación y autorización de los intercambios de datos son confiados por entidades centrales de terceros. Esto, a su vez, genera costos en términos de mantenimiento de servidores centralizados y cuellos de botella en los costos de rendimiento. No es necesario que los nodos confíen en una entidad central para que participen en transacciones entre ellos en la infraestructura basada en blockchain para mantener registros o realizar autorizaciones.

Considere una red de entidades interconectadas, acertadamente denominadas "**nodos**", que operan dentro de un sistema distribuido. Estos nodos poseen la *capacidad de intercambiar mensajes entre ellos en una forma coordenada*. Dentro de este ámbito, los nodos pueden asumir una de tres características: honesto, defectuoso o malicioso. Un nodo que exhibe una conducta errática se encuentra clasificado como un "**nodo bizantino**" o "**Byzantine node"** en inglés, cuyo nombre deriva del ilustre Problema de los generales bizantinos.

Por desgracia, el comportamiento inestable e impredecible de los nodos bizantinos plantea un desafío formidable para el funcionamiento fluido de la red. Dichos nodos, impulsados por la intención o el diseño, desencadenan consecuencias perjudiciales para la funcionalidad general del sistema. Por lo tanto, cualquier acción imprevista promulgada por un nodo, independientemente de sus motivos subyacentes, cae directamente bajo la rúbrica bizantina.

Imaginemos ahora una instancia concisa de un sistema distribuido, que comprende seis nodos, donde un nodo en particular (N4) ejemplifica la conducta bizantina, lo que lleva al posible enredo de la coherencia de los datos. Además, un enlace específico (L2) exhibe fragilidad, ya sea cortado o lento, allanando así el camino para la partición de la red y comprometiendo el flujo de información vital.

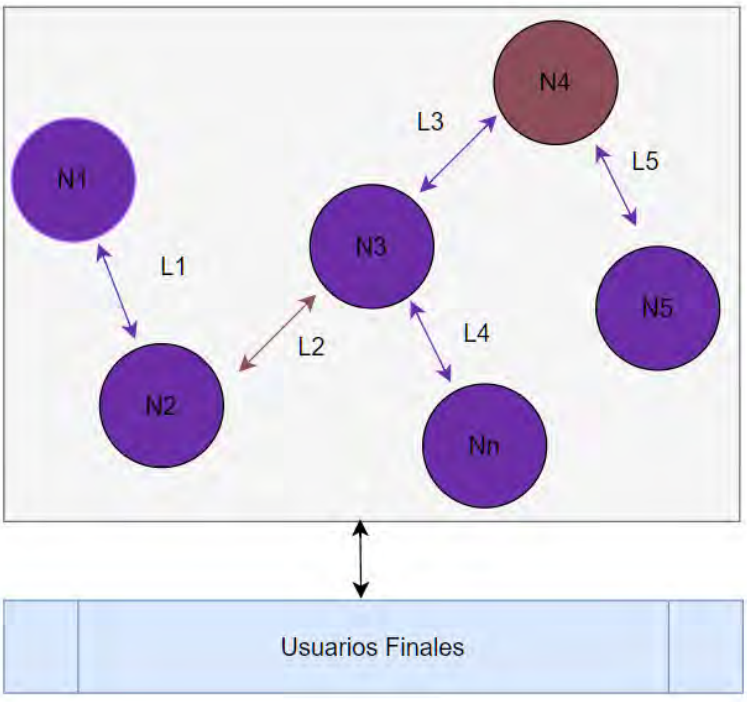

Figura 2: Diseño de un sistema distribuido: N4 es un nodo bizantino, L2 está roto o es un enlace de red lento

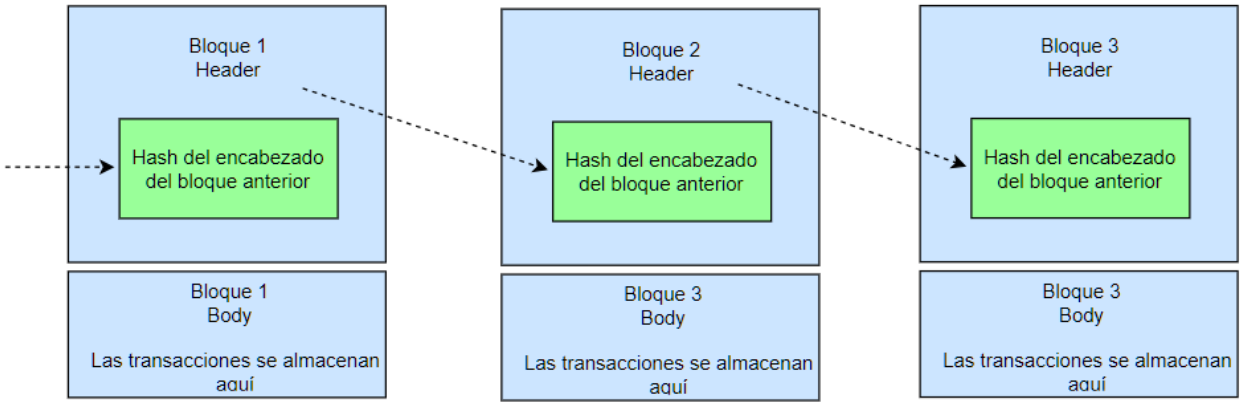

Figura 3: Representación lógica de una cadena de bloques

2.4.2 Peer-to-Peer

En peer-to-peer no hay un controlador central en la red y todos los participantes pueden hablar directamente entre sí. Muy útil y ejercitado para transacciones en efectivo ya que puede

intercambiarse directamente entre los pares sin necesidad de intervención de un tercero, como un banco.

2.4.3 Registro Distribuido (Distributed Ledger)

En términos simples, un libro o registro mayor distribuido se refiere a un sistema donde un libro mayor, o registro de transacciones, es compartido y mantenido por todos los participantes en una red. Cada participante tiene una copia completa de todo el libro mayor, lo que garantiza que la información esté ampliamente disponible y accesible en toda la red.

2.4.4 Inmutabilidad

Es indiscutiblemente difícil manipular blockchain por el hecho de que es resistente a la censura y porque todas las nuevas entradas son acordadas por pares a través de un consenso descentralizado. Para los registros, todos los contenidos modificados pueden detectarse y se dice que esos registros son inmutables. Para que un atacante altere cualquier registro, necesitaría comprometer muchos de los nodos involucrados en la red blockchain.

Tabla 1 que representa la comparación de protocolo de consenso

	PoW	**PoS**	**PoET**	**BFT y Variantes**
Tipo BC	Sin Permiso	Ambos	Ambos	Con Permiso
¿Se Requiere Token?	Si	Si	No	No
Flujo de Transacción	Alto	Alto	Medio	Alto
Finalidad de Transacción	Probabilístico	Probabilístico	Probabilístico	Inmediato
Escalabilidad de una red peer	Alto	Alto	Alto	Bajo
Costo de Participación	Si	Si	No	No

2.4.5 Auditabilidad

Para que los pares tengan acceso a todos los registros de transacciones con marca de tiempo, tienen copias de la cadena de bloques. Ayuda a poder verificar y buscar transacciones que involucren direcciones de blockchain específicas. La asociación de direcciones proporcionada por blockchain está lejos de ser entidades reales, sino que se proporciona de manera pseudoanonima. Esta es la razón por la que no es posible rastrear la dirección hasta su propietario y se puede interferir en las transacciones en las que se involucra una dirección de cadena de bloques específica, aunque las direcciones de cadena de bloques específicas pueden ser consideradas responsables.

Representación pseudoanonima será en la figura 3:

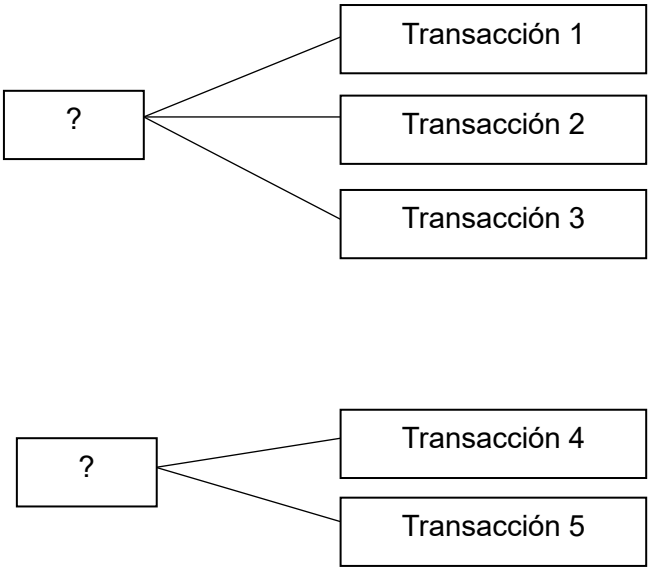

Figura 4: Situación donde a identidad de la persona no es conocida

2.4.6 Tolerancia a Fallas

Los registros del libro mayor se replican de manera idéntica en todos los pares de blockchain, en caso de fugas de datos o fallas, se pueden identificar a través de un consenso descentralizado. Una forma de resolver la fuga de datos es mediante el uso de las tiendas de réplicas en los pares de la cadena de bloques.

2.4.7 Encadenamiento de Bloques

Los bloques se generan y se rellenan con transacciones. La minería ocurre cuando el bloque se llena y luego se transmite a través de la red. Al finalizar, el bloque se encadena con el bloque anterior, que a su vez crea un libro mayor. Este proceso es infinito, lo que significa que puede repetir el proceso varias veces para crear una cadena interminable de bloques. El proceso de encadenamiento de bloques se muestra en la Figura 4.

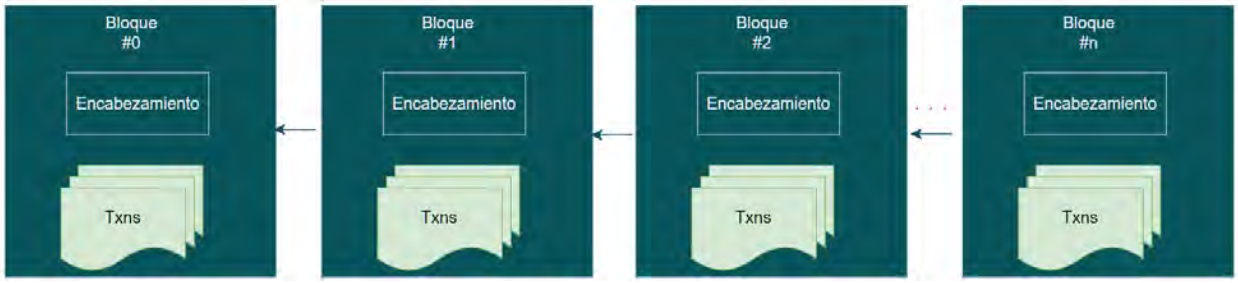

Figura 5: Block chain (Encadenamiento de Bloques)

2.4.8 Dockers y arquitectura de red

Los contenedores Docker se utilizan para configurar muchas entidades en una red Blockchain. Así, a continuación, se muestran los contenedores para las siguientes entidades en nuestra red blockchain:

- Contenedores de pares
- Contenedores Fabric CA (Certificate Authority)
- Contenedor CLI para acceder a todos los pares de la red.
- Contenedor de pedido
- Contenedores Couchdb

Todos estos contenedores son parte de una única red docker. Incluso el código de cadena se ejecuta en un contenedor separado del proceso de aprobación del par.

Capítulo III

3. TIPOS, USOS Y PLATAFORMAS DE DESARROLLO

Blockchain se puede clasificar en dos tipos:

Cadena de bloques publicas:

- Permiten que cualquier persona participe como usuario, minero, desarrollador o miembro de la comunidad. [HCD+19] Las cadenas de bloques públicas están diseñadas para estar completamente descentralizadas, sin que nadie controle cuáles las transacciones se registran en blockchain o el orden en que se procesan.

Cadena de bloques privadas:

- Blockchains privados, también conocidos como blockchains autorizados [Vuk17]. Los participantes necesitan consentimiento para unirse a las redes. Las transacciones son privadas y solo pueden acceder a ellas los participantes en el ecosistema que ha sido autorizado para unirse a la red.

3.1 COMPARACIÓN CON OTROS PLATAFORMAS BLOCKCHAINS

3.1.1 HyperLedger

Los empresarios, los administradores de TI, los empresarios, los ejecutivos y los desarrolladores de software pueden estar interesados en este tipo de blockchain, ya que son el público objetivo. Sin embargo, aunque Hyperledger es un proyecto mundial, se espera que sea utilizado por personas de todo el mundo, incluso aquellas personas que no tienen el inglés como lengua

materna. El motivo es que Hyperledger permite que una comunidad mundial de desarrolladores trabaje juntos y compartan ideas, infraestructura y código.

Hyperledger, tal como lo define Hyperledger blockchain Technologies for Business, es un esfuerzo de colaboración creado para promover a los líderes de la industria en banca, finanzas, Internet de las cosas, fabricación, suministro, cadenas y tecnología.

3.1.2 Bigchaindb

En primer lugar, bighchaindb es un sistema de código abierto. Es una combinación de características de blockchain y una base de datos distribuida de big data. Las características de blockchain consisten en la descentralización del control, el libro mayor inmutable y la gestión de la propiedad de la base de datos digital [3]. Los principales conceptos que se pueden ver en bigchaindb son:

- Los registros se agregan a BC sin Merkle Trees of sidechain.
- Las cadenas laterales permiten que los activos digitales se transfieran de forma segura entre dos BC diferentes si surge la necesidad.

3.1.3 IOTA

IOTA es una tecnología blockchain de tercera generación y es una base de datos descentralizado de última generación, y como núcleo de funcionamiento utiliza Tangle. Intercambio de información instantánea, gratuito y descentralizada son funciones que ofrece IOTA, además de realizar transacciones de valor 0 iotas (el token de IOTA). Es que por eso se ha convertido en una herramienta que se puede aplicar en el internet de las coas para el desarrollo de aplicaciones.

3.1.4 Ethereum

Pensado como una cadena de bloques programable, Ethereum es capaz de replicar y procesar datos y programas con su ecosistema de nodos (computadoras) llamados contratos en todos los nodos sin una autoridad central (Aragay, 2018) (Conteron, 2021). Las transacciones de Bitcoin y Ethereum difieren en la forma en que los usuarios de Ethereum pueden crear operaciones complejas, mientras que las operaciones de los usuarios de Bitcoin son fijas. Lo que a su vez hace que Ethereum se adapte a múltiples entornos y tecnologías, en lugar de solo criptomonedas (Kumar & Ramesh, 2019).

Tabla 2: Resumen de las diferentes clasificaciones

Características	Publica	Privada
Usabilidad	• Validaciones de documentos y operaciones IoT	• Cadena de suministro, bienes raíces y propiedad de activos.
Acceso	• Naturaleza independiente con total transparencia y confianza. • Cualquiera puede unirse	• Fully controlled Access • Requiere permisos para unirse
Transparency/ Auditability/ Security	• Falta de seguridad	• Falta de auditabilidad
Escalabilidad	• Limitado por diseño	• El diseño se puede modificar para permitir más transacciones

3.2 ALMACENAMIENTO DE INFORMACIÓN EN SISTEMA IOT

En sistemas IoT o cualquier sistema cual pueda generar datos, se puede a llegar generar enormes cantidades de estos datos, y es deseable almacenar esta información. Esta información ya almacenada se puede utilizar para análisis, mantenimiento de datos o sensores. Ante esta situación, se consideran y detallan diferentes métodos de almacenamiento para una selección adecuada para el desarrollo del proyecto.

3.2.1 Almacenamiento Local

Existen dispositivos que suelen almacenar datos internamente, sin embargo, esta práctica resulta poco recomendable. La información guardada en el dispositivo se encuentra expuesta a riesgos asociados con fenómenos ambientales que podrían dañar el hardware. En este proyecto, con el objetivo de asegurar un adecuado control de los datos, se plantea implementar un sistema de

almacenamiento remoto, y a su vez, local mediante el uso de IBeacons. El cliente encargado de recibir y gestionar los datos generados por los IBeacons será una máquina Ubuntu.

En este contexto, se busca evitar la vulnerabilidad y fragilidad inherente al almacenamiento local, optando por una solución más segura y robusta que garantice la integridad de los datos frente a posibles daños ocasionados por factores ambientales.

Las diferentes alternativas para llevar a cabo el almacenamiento se describen a continuación.

3.2.2 Archivos CSV

Una forma sencilla de presentar datos en forma de tabla es usando archivos CSV (Comma-Separated Values). La forma que se separa es usando comas. Las comas representan columnas y los saltos de línea, filas. Los archivos CSV a menudo se usan cuando los datos deben ser compatibles con muchos programas diferente que tienen métodos de traducción automática y de reconocimientos, como Excel, y para almacenar grandes cantidades de información de forma compacta.

Un ejemplo de archivo CSV:

```
Nombre,Edad,Correo,Teléfono
John Smith,33,john@gmail.com,123-456-7890
Alicia Morry,25,alicia@gmail.com,987-654-3210
Sebestian Ek,24,sebasek@gmail.com,535-247-5190
Emily Nesty,25,nesty@gmail.com,671-576-8464
```

La representación en forma tabla es:

Tabla 3:Representación del archivo CSV en una tabla

Nombre	Edad	Correo	Teléfono
John Smith	33	john@gmail.com	123-456-7890
Alicia Morry	25	alicia@gmail.com	987-654-3210
Sebestian Ek	24	sebasek@gmail.com,	535-247-5190
Emily Brown	25	nesty@gmail.com	671-576-8464

3.2.3 Base de Datos

La producción masiva de métricas de información por parte del Internet de las Cosas, necesitamos una base de datos para el manejo efectivo de esa información. Para nuestra base de datos queremos que se cumplan, por lo menos, las siguientes características:

- Escalabilidad
- Que su estructura sea flexible.
- Velocidad adecuada para gestiona enromes cantidades de información.
- Disponibilidad continua, y capacidad de tolerar fallos.
- Compatibilidad con programas o herramientas de análisis, coste y seguridad.

A continuación, se describirá diferentes tipos de base de datos lo cual se escogerá la mas adecuada para el desarrollo de este proyecto.

3.2.4 InfluxDB

Introducido en 2013, y todavía utilizado hasta el día de hoy, es una base de datos de pedidos por períodos. Se utiliza para presentar y procesar información de orden de tiempo. El acceso de esta base de datos es mediante el software Grafana, cual provee funciones de visualización para datos en series temporales.

Las ventajas de usar InfluxDB para la información de IoT incluyen:

a) Disposición de rango permitida
b) El lenguaje de consulta SQL es el mismo que InfluxDb
c) Da una introducción directa implícita a la información perdida.
d) Refuerza la información programada ahogando el examen
e) Admite consultas césales para registrar totales

3.2.5 MongoDB

MongoDB es una base de datos de código abierto NoSQL orientada a documentos que se enfoca fuertemente en el rendimiento, flexibilidad y escalabilidad. Al utilizar documentos similares a JSON con esquemas es que se clasifica como una base de datos NoSQL.

En la gestión de datos, la orientación de documentos se aparta del enfoque tradicional centrado en filas de las bases de datos relacionales. En su lugar, introduce un concepto más versátil de

un documento. Este enfoque ofrece ventajas como evitar estructuras de bases de datos rígidas y adoptar un cierto nivel de desnormalización. Al permitir la incrustación de documentos entre sí, allana el camino para mejorar el rendimiento, eliminando la necesidad de costosas operaciones de combinación.

Para interactuar efectivamente con MongoDB desde un lenguaje de programación, un intermediario, conocido como "interfaz" o "controlador (driver, en inglés)", facilita la comunicación entre el sistema de base de datos y el lenguaje de programación. Este intermediario juega un papel crucial para que la interacción sea fluida y eficiente.

3.2.6 MySQL

En el vasto ámbito de la gestión de datos, MySQL emerge como una opción popular cuando se utilizan sistemas de bases de datos de código abierto. Innumerables personas interactúan con sus capacidades, a menudo sin siquiera darse cuenta, mientras navegan por las plataformas de Facebook, Google, Amazon y el popular sitio web de conocimiento, Wikipedia. Cada vez que los usuarios emplean estas plataformas, están, de hecho, comprometiéndose con el funcionamiento de MySQL. Se ha ganado legítimamente su lugar como una opción favorita para los sitios web públicos que buscan una base de datos rápida y constante, como lo relata Oreily.

Sin embargo, no se limita a solo a las aplicaciones basadas en la web, ya que la utilidad de MySQL se extiende más allá de las aplicaciones basadas en la web. También incursiona en aplicaciones no basadas en la web.

Michael 'Monty' Widenius es quien creó MySQL. MySQL es un sistema de gestión de bases de datos relacionales que mantiene con orgullo el estandarte de los principios de código abierto, invitando a todos a beneficiarse de sus capacidades de subprocesos múltiples.

3.2.7 MariaDB

En la actualidad, MariaDB está emergiendo rápidamente como un formidable sucesor de MySQL, ya que no solo proporciona las mismas características, sino que va más allá. MariaDB es una bifurcación popular de MySQL, que satisface las demandas tanto de las necesidades empresariales como de las tareas de procesamiento de datos a pequeña escala.

Sus características clave incluyen:

- El cumplimiento total de las licencias LGPL, BSD o GPL.

- Un lenguaje de consulta (querying) aclamado y ampliamente utilizado.
- Versatilidad para ejecutarse en diversos sistemas operativos y compatibilidad con una amplia gama de lenguajes de programación.
- Amplio soporte para PHP, un lenguaje popular en el desarrollo web.
- Presume de la destreza tecnológica del Clúster Galera.

3.3 SOLUCIÓN PROPUESTA

3.3.1 Tipos de Programa: Código Abierto

Cuando hablamos de "software propietario", solo el editor y/o dueño puede modificar y ver el código del software, y está licenciado por un proveedor, normalmente por una tarifa. Por otro lado, el código abierto se refiere y ofrece la capacidad de ver y modificar el código que cualquiera puede descargar. Esta es la razón por la que se eligieron BigchainDB y Ubuntu para este proyecto.

El código abierto tiene algunos beneficios claros como los siguientes:

- Características y capacidades competitivas
- Sin bloqueo de proveedor, por lo que los clientes pueden cambiar fácilmente
- Soluciones de alta calidad
- La capacidad de personalizar y corregir errores mediante el acceso al código fuente
- Menor costo total de propiedad

3.3.2 Blockchain: BigchainDB

Se ha escogido BigchainDB por sus características idóneas para la aplicación de este proyecto. Ethereum tanto como Bitcoin carecen de utilidad en nuestro sistema propuesto. Ya que Ethereum se usa como establecimiento de contratos inteligentes y el desarrollo de otras aplicaciones blockchain. Mientras que Bitcoin es principalmente para reserva de valor u “oro digital”.

A. Bigchain DB

Esta plataforma satisface el objetivo de establecer un servidor descentralizado para registrar la información recogida. De esta manera la información va a quedar escrita de forma íntegra e inmutable y sea fácilmente consultada. Además, que proporciona adicionalmente mongoDB como base de datos.

3.3.3 Base de Datos: MariaDB

Se va a utilizar la base de datos tipo MySQL para el almacenamiento de los datos recogidos por el sensor IBeacon conectado al Raspberry Pi. Específicamente se utilizará MariaDB por su API de base de datos de Python, y su interfaz compatible con subprocesos para el servidor de MySQL. También porque sigue un criterio 100% Open Source y libre, no requiere ningún tipo de licenciamiento, y es muy simple de instalar.

3.4 COMO BIGCHAINDB FUNCIONA

La compresión de las estructuras de datos es crucial cuando se trabaja con BigchainDB. A diferencia de las bases de datos SQL tradicionales donde los datos están estructurados en tablas, BigchainDB organiza los datos por activos (assets). Además, las bases de datos NoSQL como JSON permiten varios formatos de estructuración de datos.

3.4.1 Transacciones BigchainDB

Una transacción de BigchainDB es una cadena JSON que se ajusta a una especificación de transacciones de BigchainDB (BigchainDB GmbH, 2018).

La especificación de cada transacción proporciona información detallada sobre las claves requeridas y sus valores correspondientes, junto con sus significados. Incluye instrucciones paso a paso sobre cómo crear una transacción, un conjunto de comprobaciones que deben realizarse para validar la corrección de una transacción y detalles específicos sobre las primitivas criptográficas utilizadas (BigchainDB GmbH, 2018). A continuación, se muestra una ilustración de una transacción BigchainDB (v2) como se muestra en la figura 6.

3.4.2 Transacciones CREATE

Para registrar, crear o iniciar el historial de un activo en BigchainDB, se utiliza una transacción CREATE. Entidades o trabajos creativos se pueden registrar o un cualquier tipo, y estos objetos registrados se denominan activos.

3.4.3 Transacciones TRANSFER

Una transacción TRANSFER puede transferir uno o más salidas (outputs) en otras transacciones. Las salidas deben estar asociadas al mismo activo y una transacción de TRANSFER solo puede transferir acciones de un activo a la vez.

3.4.4 Componentes de una Transacción

Una transacción tiene la siguiente estructura básica:

```
{
      “id”: id,
      “version”: version,
      “inputs”: inputs,
      “outputs”: outputs,
      “operation”: operation,
      “asset”: asset,
      “metadata”: metadata
}
```

- ID de la transacción
  El ID es una cadena y corresponde al SHA3-256 de la transacción cual tiene la forma: "0e6a3h5047fdf39ag5ead7171eb412g6bffdbe4d7428966584b4147853d19976"

- Inputs
  Se puede conceptualizar el input como un puntero a un output de una transacción previa. Una lista de entradas de transacciones es el valor de "entradas". Cada entrada de transacción transfiere una salida de transacción anterior. Una transacción CREATE debe tener exactamente una entrada (es decir, num_inputs == 1).

  Estructura básica de “inputs”

  ```
  {
     "fulfills": {
        "transaction_id": transaction_id,
        "output_index": output_index
     },
     "owners_before": [public_key_1, public_key_2, etc.],
     "fulfillment": fulfillment
  }
  ```

- Output

El valor de “outputs” es una lista de transacciones. Cada salida indica las condiciones criptográficas que deben cumplirse por cualquier persona que desee transferir/gastar esa salida. También indica la cantidad de participaciones del activo vinculado a esa salida.

Estructura básica de “outputs”

```
{
    "condition": condition,
    "public_keys": [public_key_1, public_key_2, etc.],
    "amount": amount
}
```

- Operation
  “Operation” indica el tipo de transacción de la que se trata y como debe ser validada. Es una cadena y los valores admitidos son “CREATE” y “TRANSFER”.

- Asset

  Es una transacción del tipo CREATE, un activo puede ser nulo o una matriz asociativa que contiene exactamente un par clave-valor. De tipo “datos” la llave tiene que ser y el valor puede ser cualquier matriz asociativa válida.

Ejemplo de JSON

```
{
    "data": {
        "desc": "Aston Martin",
        "Model": " DBS 770 Ultimate",
        "Asset type": "Car",
        "Owner": "Dyel Acosta"
    }
}
```

- Metadata

Permite al usuario agregar información cualquier tipo de dato adicional a la transacción. Se actualiza por cada transacción que se realiza.

Ejemplo de JSON


```
{
    "coordinates": "13 23 23",
    "humidity": "wet",
    "location": {
        "name": "Laboratorio de Redes",
        "size": "3",
        "nodes": "3"
    }
}
```

```
{
    "id": "3667c0e5cbf1fd3398e375dc24f47206cc52d53d771ac68ce14d c,
      → df0fde806a1c",
    "version": "2.0",
    "inputs": [
        {
            "fulfillment": "pGSAIEGwaKW1LibaZXx7_NZ5-V0alDLvrguGLyL c
            → RkgmKWG73gUBJ2Wpnab0Y-4i-kSGFa_VxxYCcctpT8D6s4uTGOO c
            → F-hVR2VbbxS35NiDrwUJXYCHSH2IALYUoUZ6529Qbe2g4G",,
    ,
    "fulfills": null,
            "owners_before": [
                "5RRWzmZBKPM84o63dppAttCpXG3wqYqL5niwNS1XBFyY"
            ]
        }
    ],
    "outputs": [
        {
            "amount": "1",
            "condition": {
                "details": {
                    "public_key": ,
                     → "5RRWzmZBKPM84o63dppAttCpXG3wqYqL5niwNS1XBFyY",
                    "type": "ed25519-sha-256"
                },
                "uri": "ni:///sha-256;d-_huQ-eG-QQD-GAJpvrSsy7lLJqyNh c,
                 → tUAs_own7aTY?fpt=ed25519-sha-256&cost=131072"
            },
            "public_keys": [
                "5RRWzmZBKPM84o63dppAttCpXG3wqYqL5niwNS1XBFyY"
            ]
        }
    ],
    "operation": "CREATE",
    "asset": {
        "data": {
            "message": "Greetings from Bel!"
        }
    },
    "metadata": null
}
```


Figura 6: Transacción en BigchainDB

3.4.5 Enviar una Transacción en una Red de BigchainDB

Consultas (Querying) BigchainDB

La consulta de BigchainDB se facilita a través de MongoDB, que permite al operador del nodo buscar y recuperar datos almacenados. El operador tiene acceso completo al MongoDB local del nodo BigchainDB. Para ejecutar consultas, se pueden utilizar controladores MongoDB como Mongo Shell, que brindan acceso a las API de MongoDB.

La conexión a MongoDB

Para consultar primero debemos conectarnos a la base de datos MongoDB. La conexión se realiza mediante un nombre de host y un puerto. El nombre de host depende de dónde se encuentre el nodo BigchainDB, si está en una máquina local, entonces el nombre de host debe ser *localhost* y con el puerto 27017.

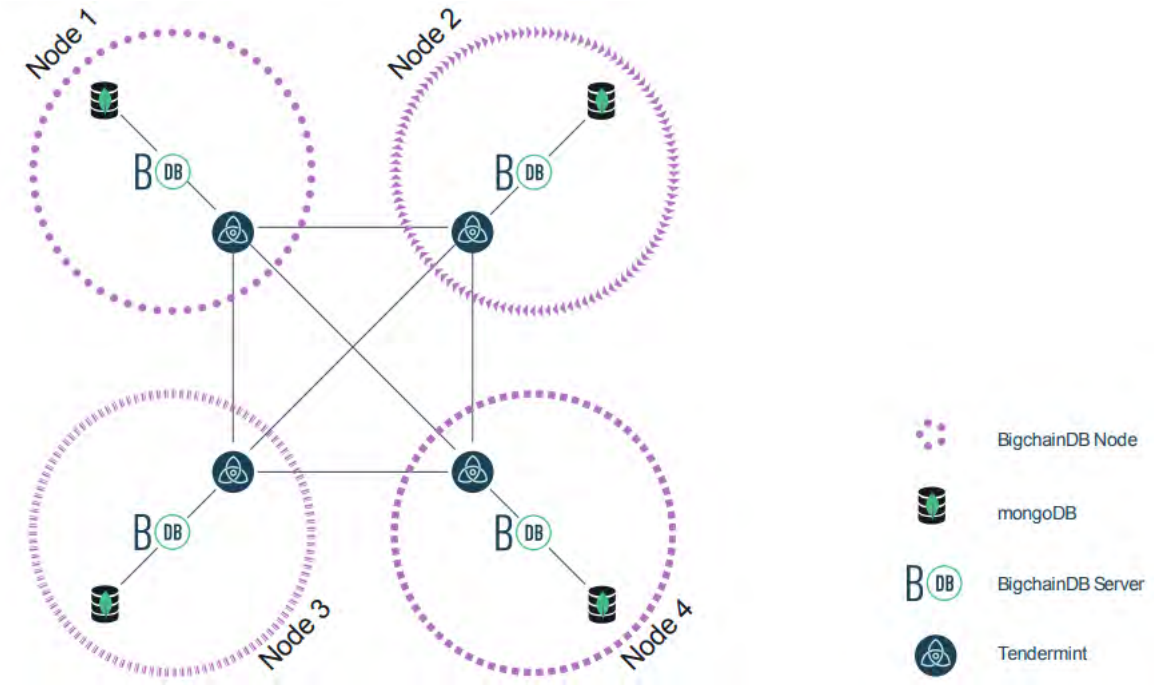

Figura 7: Red de 4 nodos de BigchainDB. Fuente http://docs.bigchaindb.com/en/latest/installation/api/http-client-server-api.html

Capítulo IV

4. DESARROLLO DEL SISTEMA

4.1 DESCRIPCIÓN DE LA ARQUITECTURA GENERAL DEL SISTEMA PROPUESTO

El sistema está formado por un ordenador de placa única Raspeberry Pi, sensores iBeacons conectados mediante bluetooth a dicha placa, un servidor local con sistema operativo Ubuntu 18.04.6 LTS cual tiene la base de datos MariaDB, el blockchain BigchainDB y IOTA al que publicara un resumen estadístico temporal periódico de dichas mediciones y que permite realizar un control preciso a tiempo real de las condiciones ambientales requeridas.

- El sistema se desarrollará en lenguaje de programación de Python, pues resulta idóneo para la interacción y la creación de aplicaciones en la Raspberry Pi.

- Un sensor iBeacon, compatible con la Raspberry Pi, utilizando mediciones y capaz de medir el posicionamiento de objetos.

- El protocolo de comunicación entre maquinas Message Queue Telemetry Transport o MQTT, que tiene una estructura bróker-cliente y servirá como canal de intercambio de mensajes entre la Raspberry Pi y con la maquina Ubuntu desde que se tratará la información recogida y se insertará en los destinos servidores. Este resulta un protocolo extremadamente ligero y por tanto ideal en este caso.

- La base de datos MariaDB, gratuita y de código abierto, como servidor local en el que registrara las los valores RSSI (Received Signal Strength Indicator) de los nodos de la red de sensores inalámbricos.y enviadas por la Raspberry Pi, siendo la primera capa de solución.

- La base de datos descentralizada y distribuida con características de blockchain que es BigchainDB, para llevar a cabo las mismas acciones que en MariaDB. Este software se vale de servidores MongoDb que se configuran en cada nodo participante de la red, y del

protocolo de comunicación blockchain Tendermint para realizar las transacciones entre dichos nodos.

4.2 TOPOLOGÍA DE LA PRACTICA:

En la figura 8 se ve la topología para implementar la tecnología blockchain en un sistema de posicionamiento de objetos en espacios interiores.

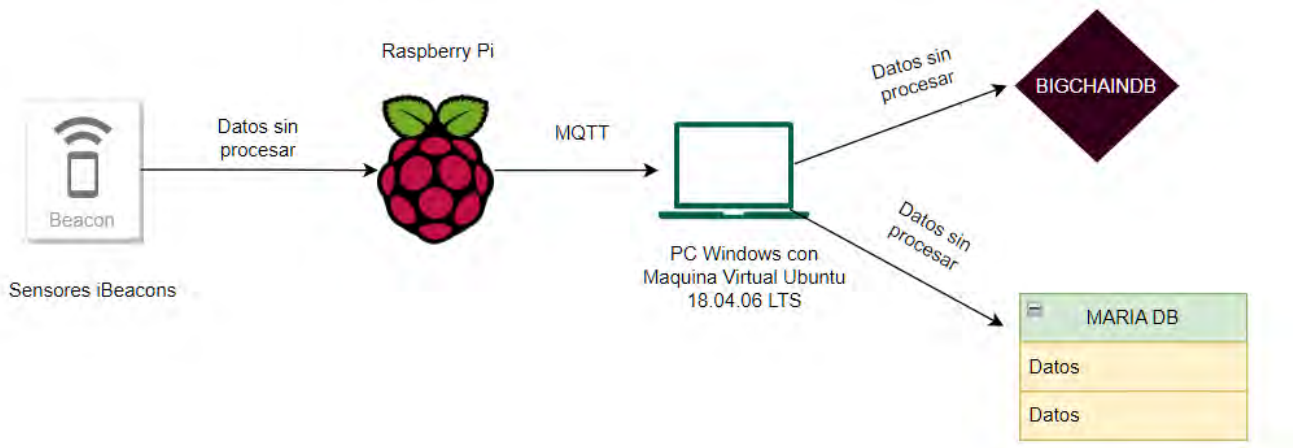

Figura 8: Arquitectura del sistema

4.2.1 Arquitectura de la distribución:

1. Capacitación de los datos
   A partir de los sensores unidos a través de bluetooth con la placa Raspberry Pi. Para ello se desarrolla un programa en el lenguaje de programación Python 3.5 que arroja datos de coordenadas.

2. Envío de los datos a máquina Ubuntu
   A través del protocolo de comunicación M2M de MQTT. Se establecerá una conexión servidor-cliente y se procederá al envió de los datos de forma periódica.

3. Recepción de los datos
   En la maquina Ubuntu, previamente suscrita al canal MQTT correspondiente. El programa cliente, una vez establecida satisfactoriamente la conexión con el bróker y recibidos los datos, realizara 3 acciones simultaneas por cada mensaje recibido (pasos a, b y c).

a. Inserción en base de datos MariaDB
b. Inserción en base de datos blockchain BigchainDB

4.3 INSTALACIÓN DE UBUNTU 18.04.6

4.3.1 Instalación del Servidor con Sistema Operativo Ubuntu 18.04.6 LTS como una Máquina Virtual en Virtual Box

Se descarga el ISO de Ubuntu 18.04.6 LTS del sitio oficial: https://www.virtualbox.org/ luego se instala en Virtual Box.

En la figura 9 se muestra la máquina virtual Ubuntu 18.04.6 LTS que va a actuar como el servidor que va a contener la base de datos MySql Mariadb y Bigchaindb, junto con Mongo db y Tendermint.

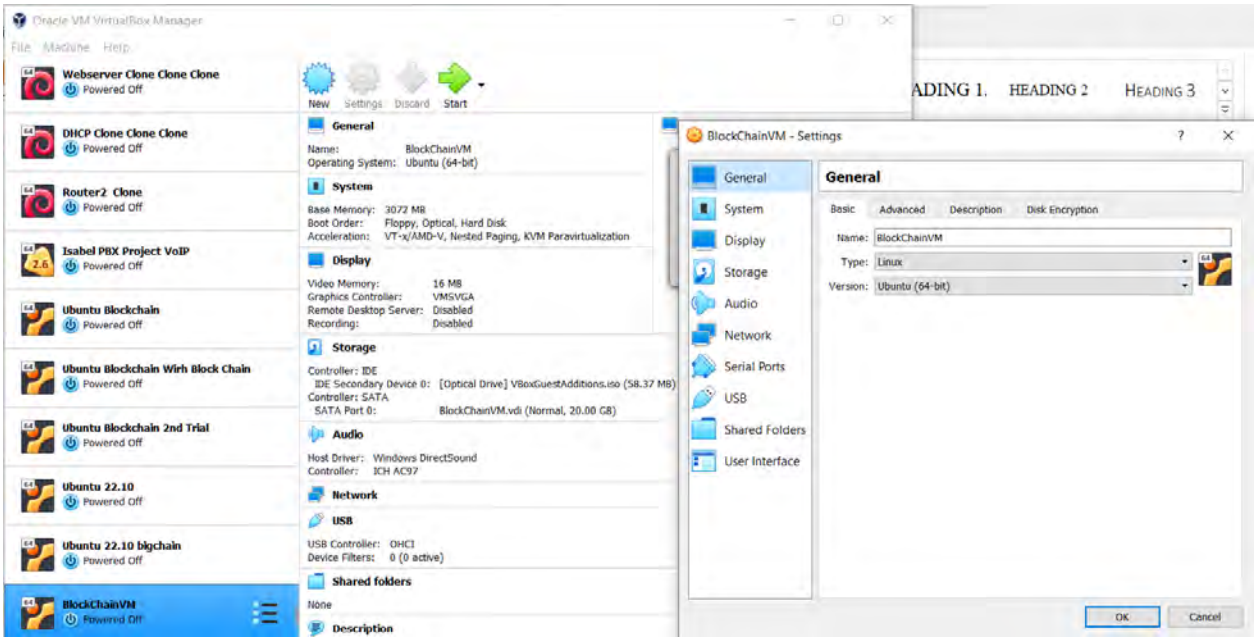

Figura 9: Máquina virtual Ubuntu 18.04.6 LTS (BlockChainVM) en Virtual Box

4.3.2 Apariencia Ubuntu 18.04.6 LTS

Una vez instalado correctamente, debería ver el inicio como la figura 10. El nombre de usuario yo le puse como Block Chain.

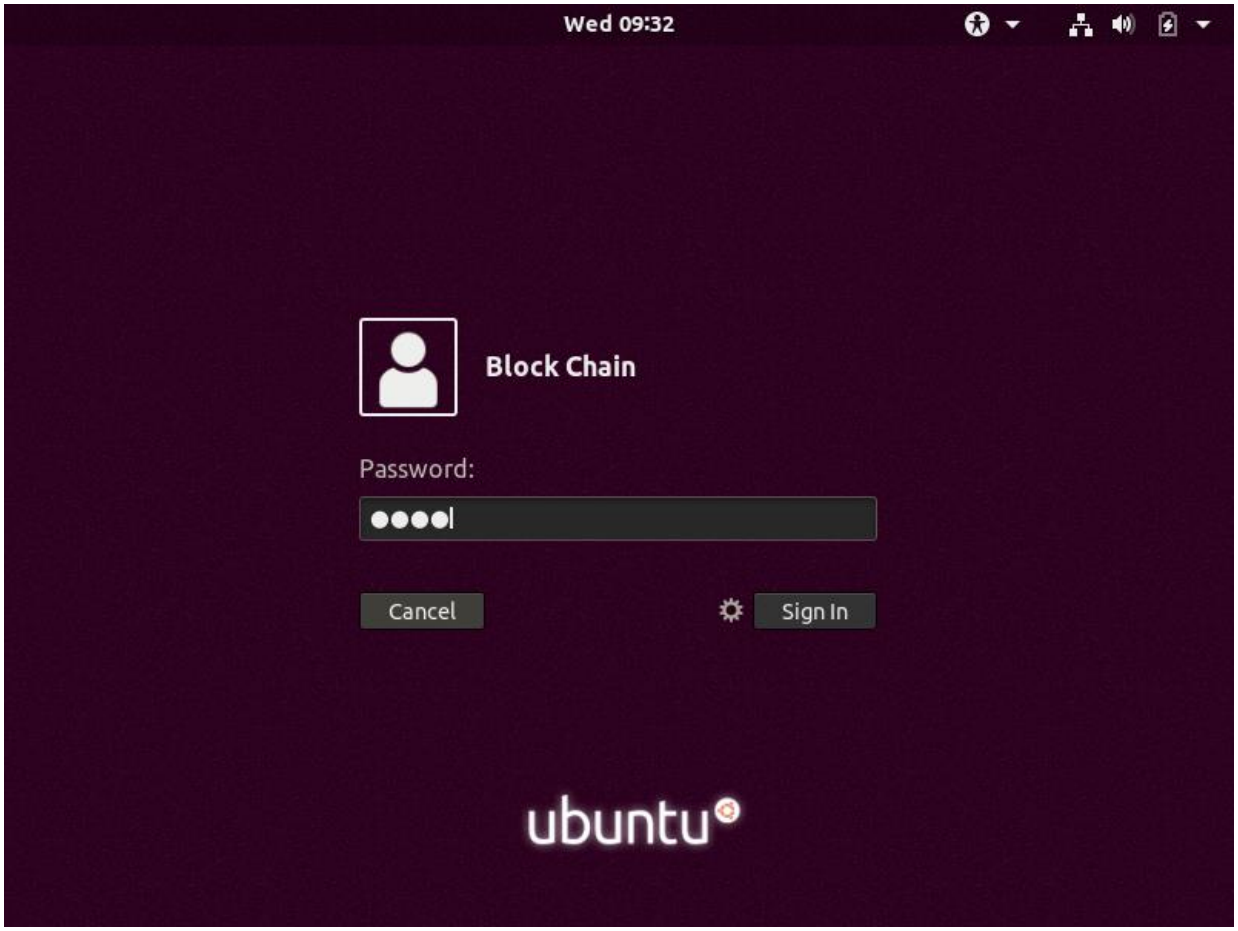

Figura 10: Inicio del Sistema Operativo Ubuntu con Nombre se Usuario Block Chain

4.4 INSTALACIÓN DE BLOCKCHAIN: BIGCHAINDB Y DEPENDENCIAS REQUERIDAS

Para instalar BigchainDB tenemos que instalar varios repositorios y software como all-in-one Docker, Tendermint, Mongodb y Bigchaindb, controladores y actualizar los repositorios actuales. Esto para que se puedo configurar el software principal. En esta sección se realiza la instalación paso a paso en orden *cronológico*:

1. *Sudo apt update*
2. *sudo apt install -y build-essential libssl-dev python3-dev python3-pip libffi-dev*
3. *sudo apt install python3-pip.*
4. *pip3 install bigchaindb_driver*

5. *pip3 install --upgrade pip setuptools*

4.4.1 Sudo Apt Update

Primero tenemos que ejecutar el comando *sudo apt-get update* o *sudo apt update* para actualizar los archivos de índice del paquete en el sistema. Además de contener los paquetes disponibles y la versión respectiva, se descargará automáticamente la versión más reciente del paquete que encontró.

4.4.2 Python3 Dev

Para construir y compilar software a partir del código fuente, tenemos que instalar herramientas y librerías de desarrollo esenciales. El código para hacerlo es: *"sudo apt install -y build-essential libssl-dev python3-dev python3-pip libffi-dev".* Esta línea de código, como se muestra en la figura 11, también instalará paquetes de Python usando pip, que serán necesarios para tareas de desarrollo y compilación de código para diferentes lenguajes de programación.

4.4.3 Pip3

Para que tengamos una gestión de paquetes para se puede utilizar para instalar y administrar paquetes de software escritos en Python, tenemos que instalar pip3 con la siguiente línea de comando: *sudo apt install python3-pip.* Para verificar la versión se ejecuta: pip -V

4.4.4 Instalación del driver de bigchaindb-driver

Se debe instalar el driver de bigchaindb con el siguiente comando: *pip3 install bigchaindb_driver*

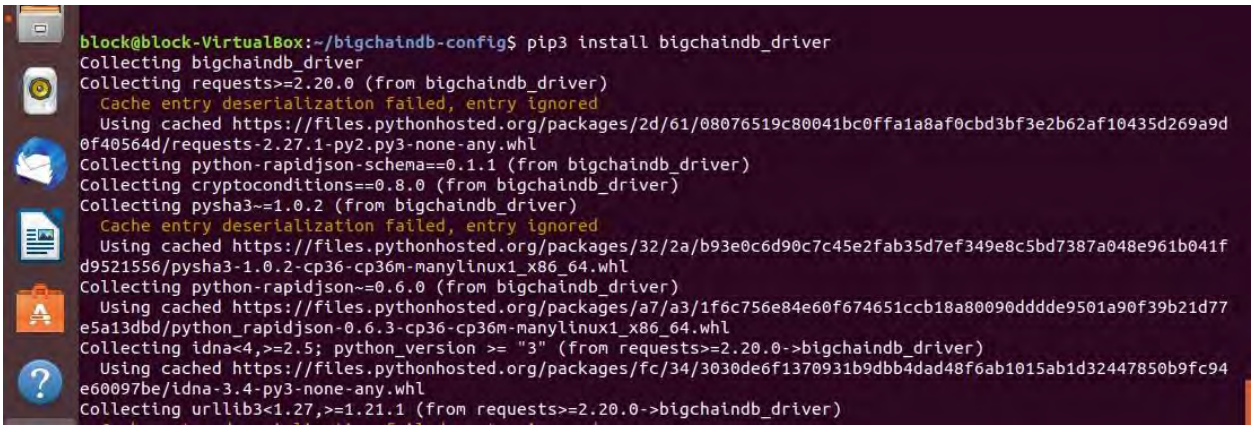

Figura 11: Instalación del driver de bigchaindb

4.4.5 Setup Tools

El software o los paquetes nuevos, como BigchainDB, por veces requieren una versión más nueva de python, por lo que, para tener una versión actualizada de python, puede actualizar pip (pip3) y setuptools a las últimas versiones usando: *pip3 install --upgrade pip setuptools*

4.5 SERVICIO DE DOCKER COMPOSE

Necesitamos una plataforma que nos permita automatizar el despliegue y la gestión de aplicaciones en contenedores portátiles ligeros. Un Docker, es esa plataforma, que nos otorgará esa automatización. *sudo apt install -y docker.io*: este comando instala Docker en el sistema. **BigchainDB** se implementará y ejecutará dentro de un contenedor Docker, proporcionando aislamiento y fácil administración.

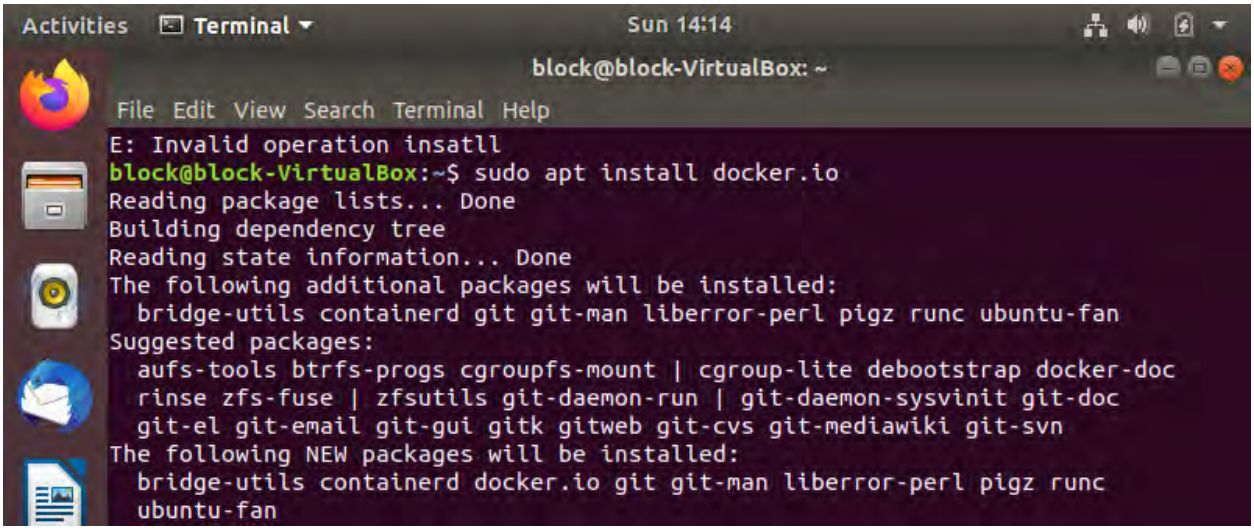

Figura 12:. sudo apt install -y docker.io: este comando instala Docker en el sistema.

4.5.1 Habilitar Docker Compose

Docker debe estar ejecutándose para administrar y crear contenedores para BigchainDB, por lo que lo iniciamos haciendo*: sudo systemctl start docker* y *sudo systemctl enable docker*. Además, cada vez que iniciemos el sistema, se iniciará automáticamente.

.

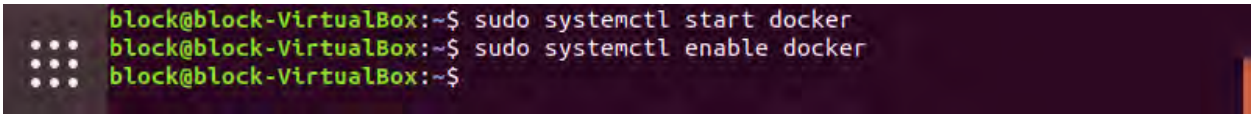

Figura 13: Inicio y habilitación del servicio Docker

Una vez que ejecutemos los comandos tenemos que verificar que efectivamente el servicio de Docker está corriendo y activo.

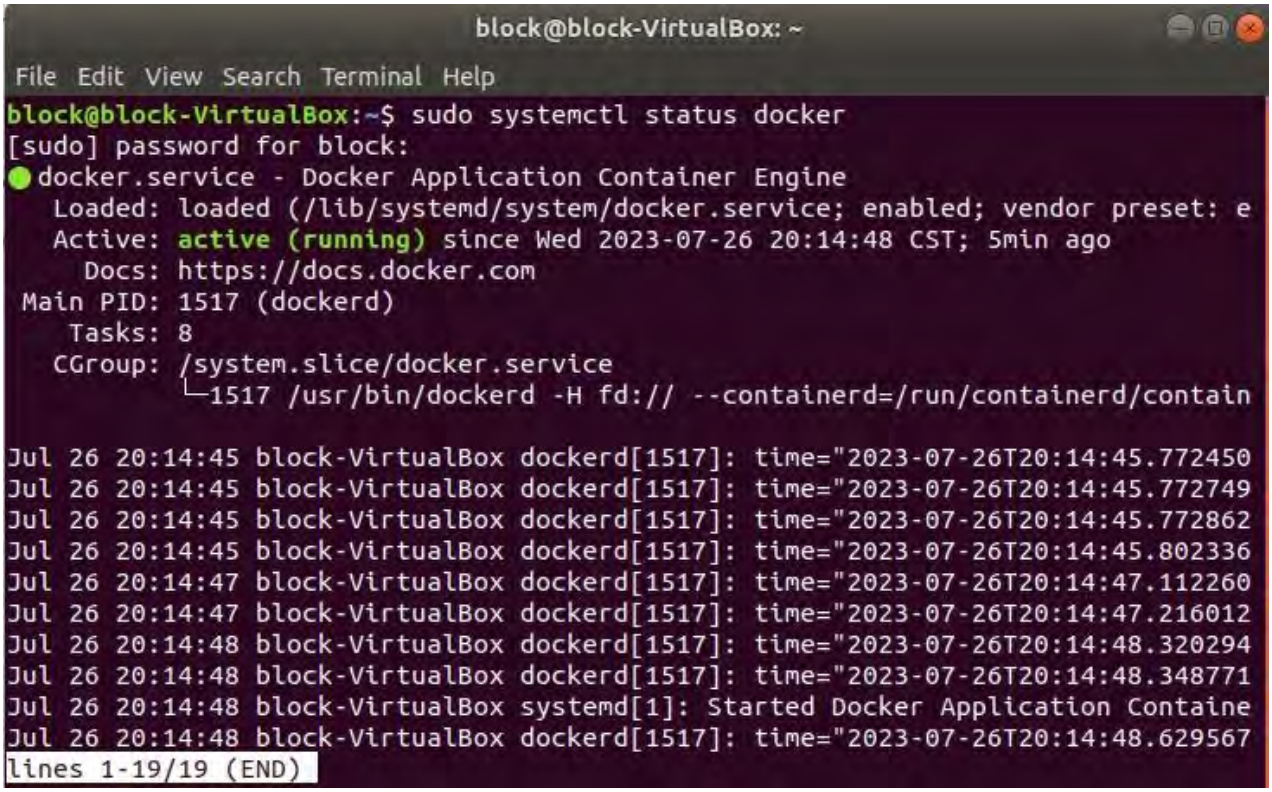

Figura 14: Estado del servicio de docker-compose

4.5.2 Instalación de Curl

Necesitaremos una herramienta que nos permita solicitar y transferir datos a través de una URL bajo diferentes protocolos. Lo haremos con cURL, lo instalamos con la siguiente línea de código: *sudo apt install curl.*

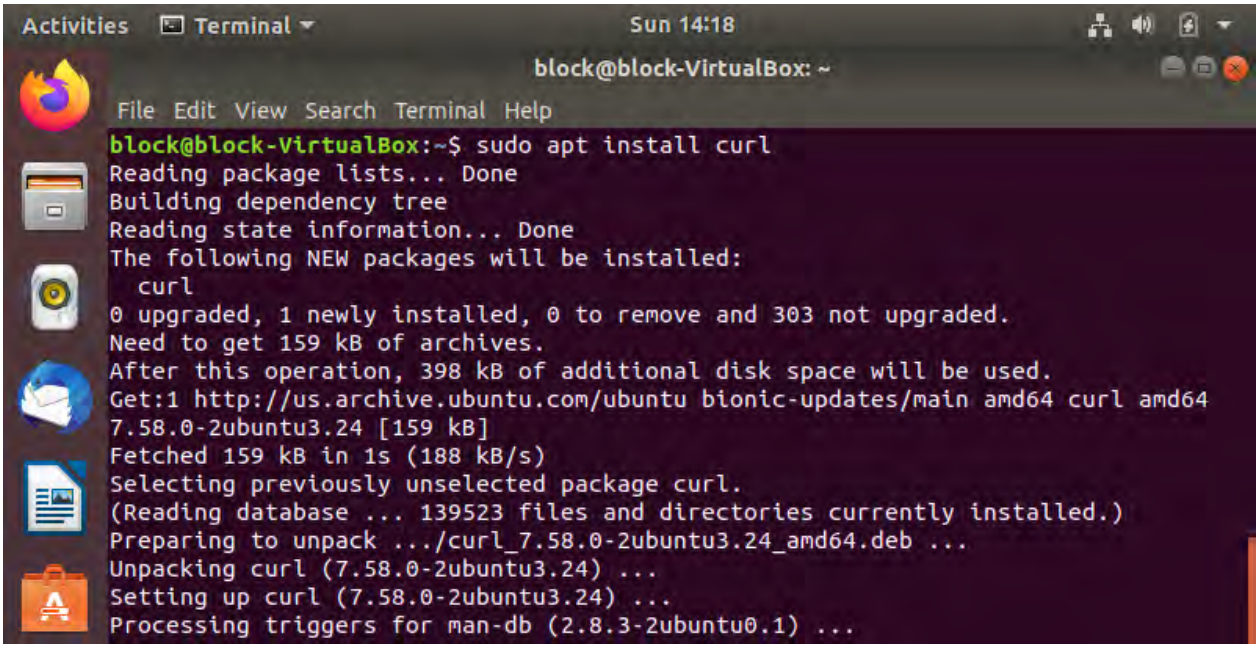

Figura 15: Instalación de Curl

4.5.3 Levantar nodo de BigchainDb a través de Docker-Compose

Tenemos que instalar una versión actualizada de Docker-Compose con el primer comando luego ejecturar el segundo comando:

- *sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/ docker-compose*
- *sudo chmod +x /usr/local/bin/docker-compose*

Estos comandos descargan el binario de Docker Compose y le otorgan permisos de ejecución. Docker Compose es una herramienta para definir y administrar aplicaciones Docker de múltiples contenedores, lo que facilita la configuración y ejecución del servicio BigchainDB junto con sus dependencias.

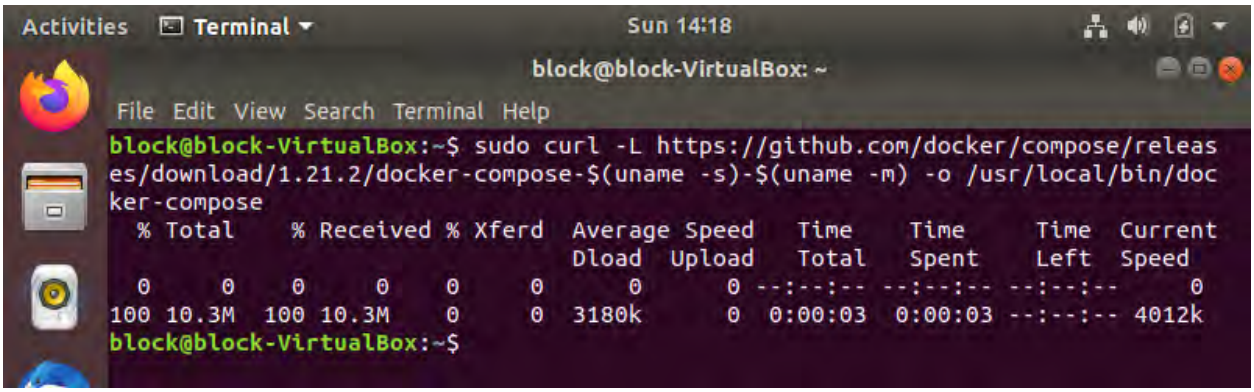

Figura 16: Instalación de Docker-Compose

Carpeta para los archivos de BigchainDB

Necesitamos crear directorios para configurar los ajustes de BigchainDB, como opciones de base de datos, ajustes de nodos y configuraciones de red. *mkdir bigchaindb-config* y *cd bigchaindb-config*: estos comandos crean un directorio y procede a acceder al directorio.

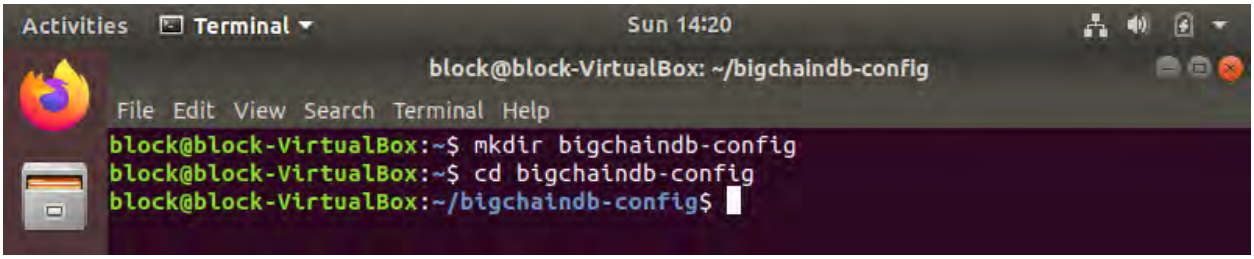

Figura 17: Creación del Directorio bigchaindb-config

Figura 18: Carpeta bigchaindb-config en Home, creado mediante el comando: mkdir bigchaindb-config

4.5.4 Creación de llaves públicas y privadas:

Necesitamos tener llevas públicas y privadas que se van a utilizar entre los nodos. Se crea un archivo de Python llamado “generate_keys.py” y se pone las siguientes líneas de códigos:

```
from bigchaindb_driver.crypto import generate_keypair

public_key, private_key = generate_keypair()

print(f"Public Key: {public_key}")
print(f"Private Key: {private_key}")
```

Figura 19: Código para obtener las llaves

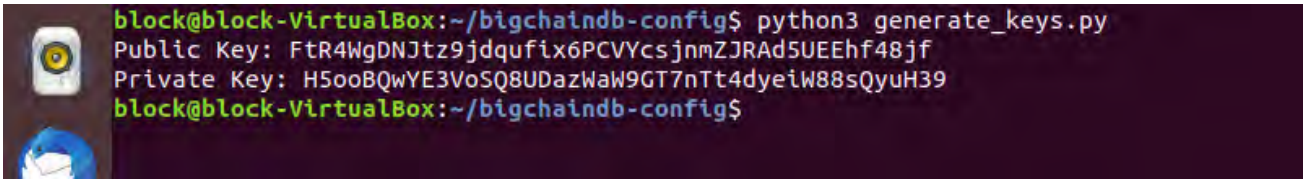

Figura 20: Ejecución del archive para obtener las llaves

Creación de Docker-compose.yml y contenido archivo docker-compose.yml. Este archivo define los servicios, las redes y los volúmenes necesarios para ejecutar el contenedor BigchainDB. Luego de ejecutar nano docker-compose tenemos que llenar el archivo con la siguiente información (solo con espacios, sin tabulaciones) como se muestra en el código siguiente:

Código del archivo YML para Tendermint, Bigchaindb y MongoDB

```yaml
version: '3'
services:
  bigchaindb:
    image: bigchaindb/bigchaindb
    volumes:
      - bigchaindb_data:/data/db
      - ./config.json:/etc/bigchaindb/config.json
    ports:
      - "9984:9984"
      - "9985:9985"
    environment:
      BIGCHAINDB_DATABASE_HOST: mongodb
      BIGCHAINDB_DATABASE_BACKEND: localmongodb
    networks:
      - bigchaindb_net

  mongodb:
    image: mongo:latest
    volumes:
      - mongodb_data:/data/db
    networks:
      - bigchaindb_net

  tendermint:
    image: tendermint/tendermint:v0.34.9
    ports:
      - "26656:26656"
      - "26657:26657"
    volumes:
      - tendermint_data:/tendermint
    networks:
      - bigchaindb_net

volumes:
  bigchaindb_data:
    driver: local
  mongodb_data:

  tendermint_data:

networks:
  bigchaindb_net:
    driver: bridge
```

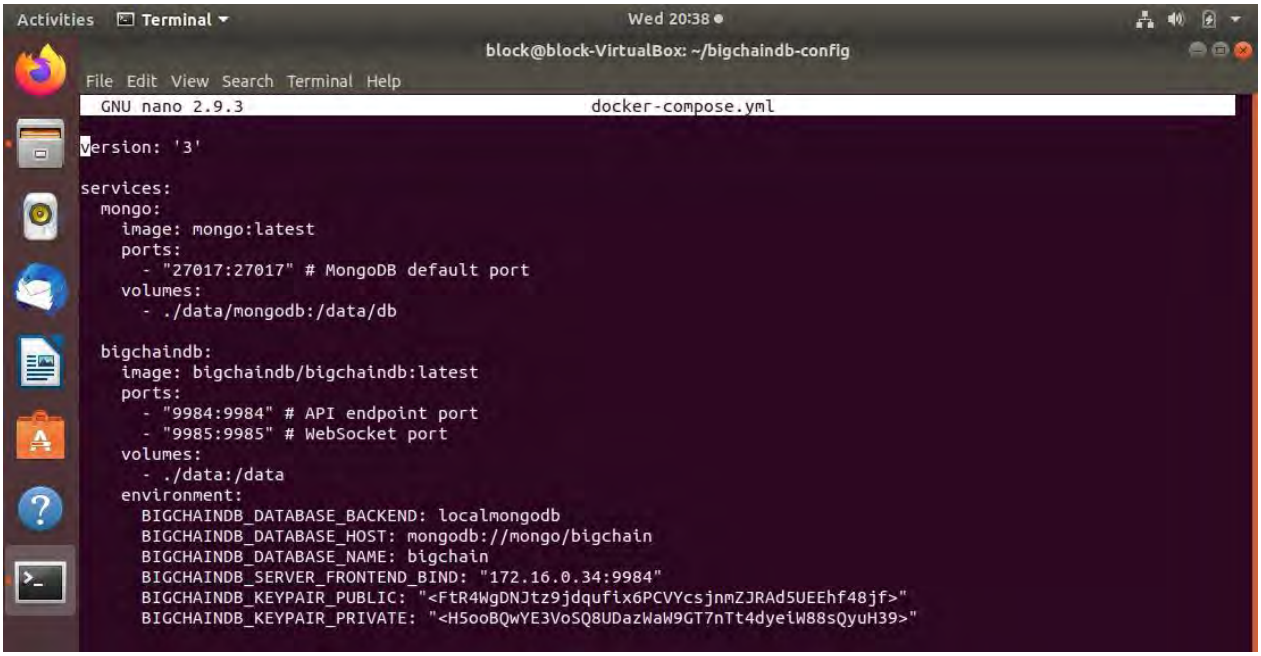

Figura 21: El contenido que debe contener el archivo para que se puede ejecutar el contendor BigchainDB

4.5.5 Crear imagen de Docker

Luego, tenemos que crear la imagen de Docker para el servicio bigchaindb en función de las configuraciones en docker-compose.yml. La imagen resultante contiene todos los componentes necesarios para ejecutar BigchainDB con la configuración especificada. *docker-compose build bigchaindb*: este comando crea la imagen de Docker

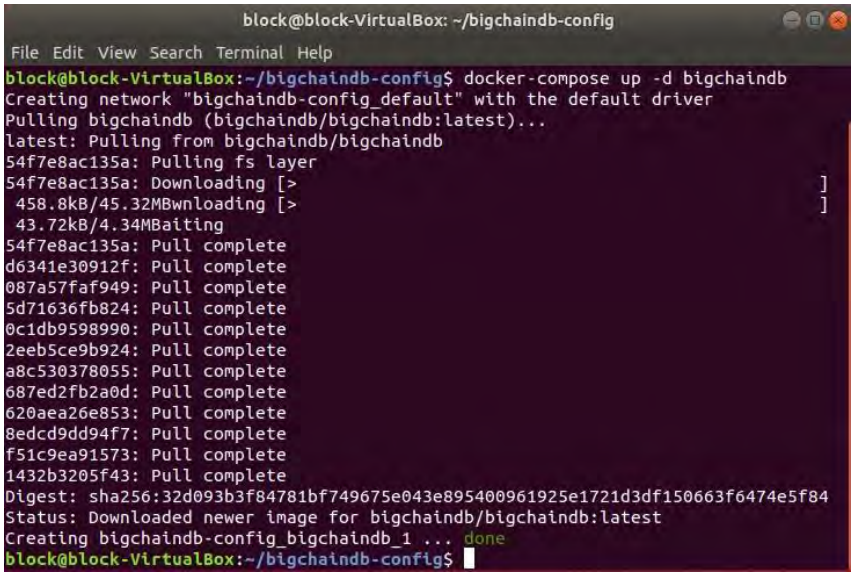

Figura 22: Creación de la imagen de Docker

4.5.6 Iniciar el servicio bigchaindb

docker-compose up -d: este comando inicia el servicio bigchaindb en modo separado (en segundo plano) utilizando la imagen de Docker creada. El nodo BigchainDB ahora se ejecutará en un contenedor Docker, listo para aceptar solicitudes de API y participar en operaciones de blockchain.

Figura 23: Inicio del servicio de bigchaindb

4.5.7 Estado de los servicios y api

Una vez que se ejejcuto los comanods sin errores podemos verificar el estado con el comando: *docker-compose ps* de igual manera se puede ingresar mediante un navegador web por el api: http://localhost:9984/api/v1/ o por http://localhost:9984.

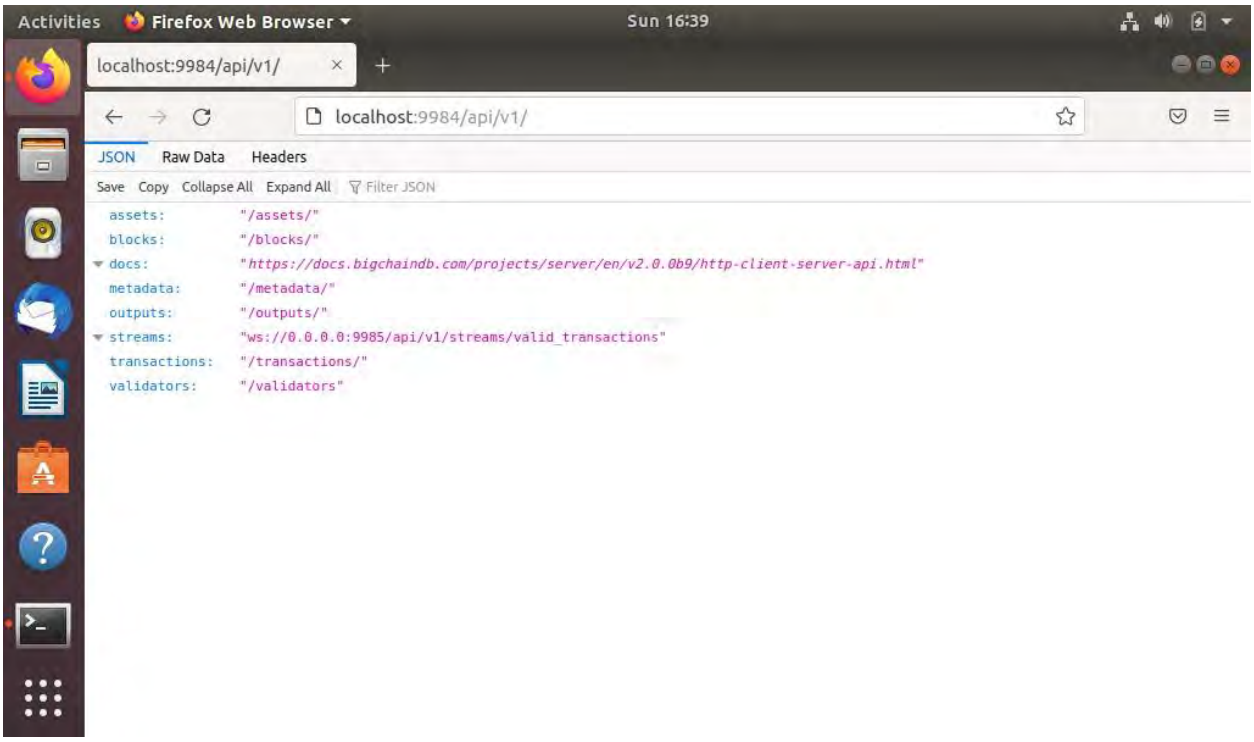

Figura 24: Acceso a Bigchaindb mediante el API

4.6 Instalación y Configuración la Base de Datos MariaDB

De primero se va a tener que instalar unos repositorios y tendremos que configurar un usuario y contraseña específico. Luego se podrá acceder al servidor con la dirección: http://localhost/phpmyadmin y visualizar la base de datos a utilizar en el proyecto.

4.6.1 MySQL

Se tiene que instalar MySQL y otros paquetes en la máquina. Luego se va a tener que instalar el módulo MySQLdb con interfaz Python para la base de datos. Los comandos para instalar MySQL y el módulo MySQLdb son:

```
sudo apt-get install mysql-server
sudo apt-get install python-mysqldb
```

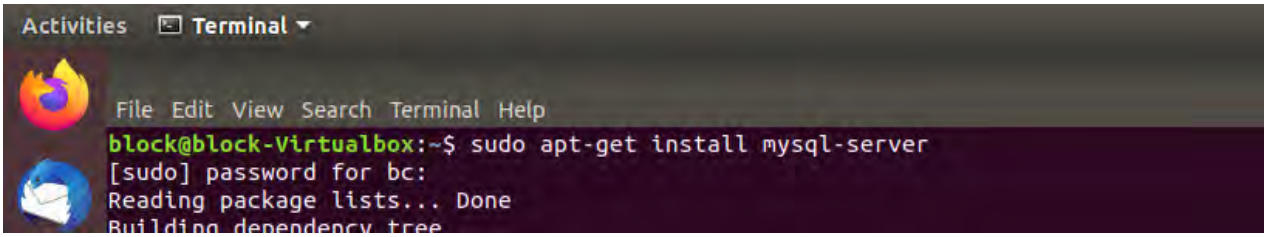

Figura 25:Instalación de Mysql Server

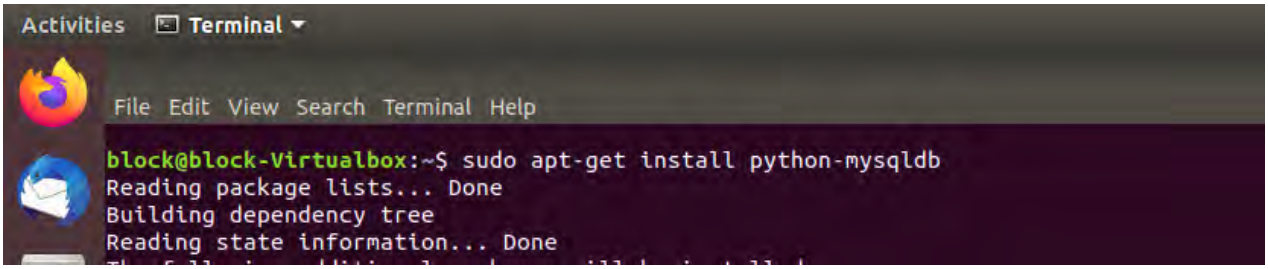

Figura 26: Instalación de Mysqldb

La versión que se instalo es: 14.14. Se puede visualizar con el siguiente comando:

mysql --version.

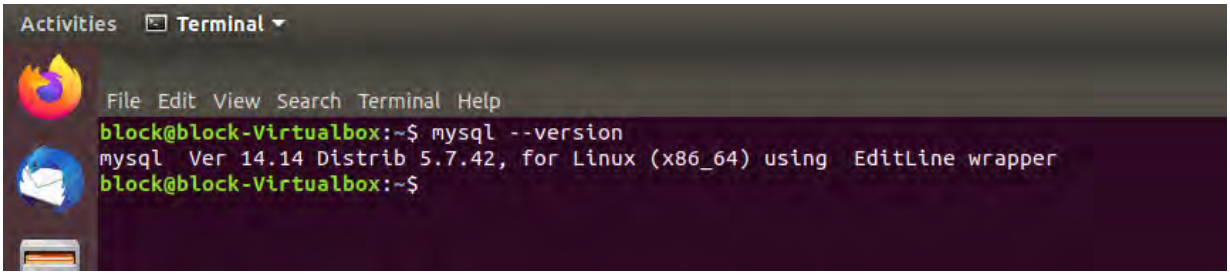

Figura 27: Version de Mysql

4.6.2 Actualización de Contraseña

Tenemos que configurar un script para la configuración de seguridad para establecer una contraseña para las cuentas raíz y para la seguridad de acceso remoto. Para poder entrar y configurar MariaDB se utiliza el comando:

```
sudo mysql_secure_installation
```

Se va a tener que seleccionar un numero entre el 0 a 2. Ya que seleccione el numero proceda a introducir su contraseña que desea utilizar para MariaD B.

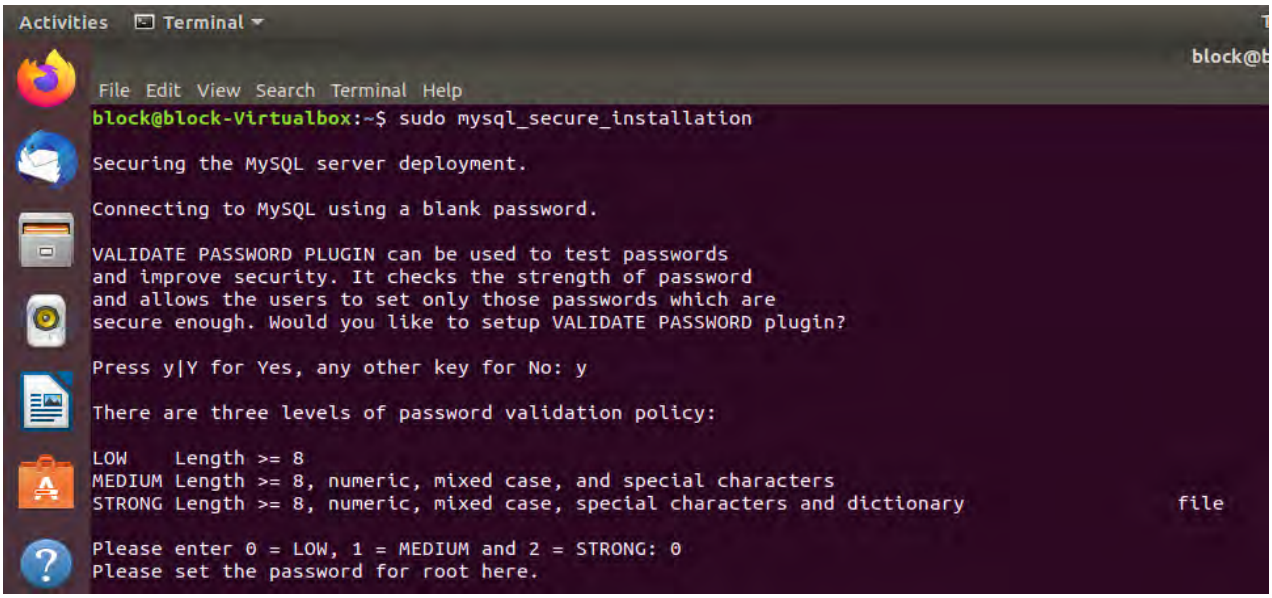

Figura 28: Interfaz para cambiar contraseña

Ahora tendremos que agregar entrar a mysql mediante el comando:

```
sudo mysql
```

Y agregue la siguiente línea de comando, en contraseña se debe de poner la contraseña que se puso en la figura 28.

```
ALTER USER ‘root’@’localhost’ IDENTIFIED WITH mysql_native_password BY
‘contraseña’
```

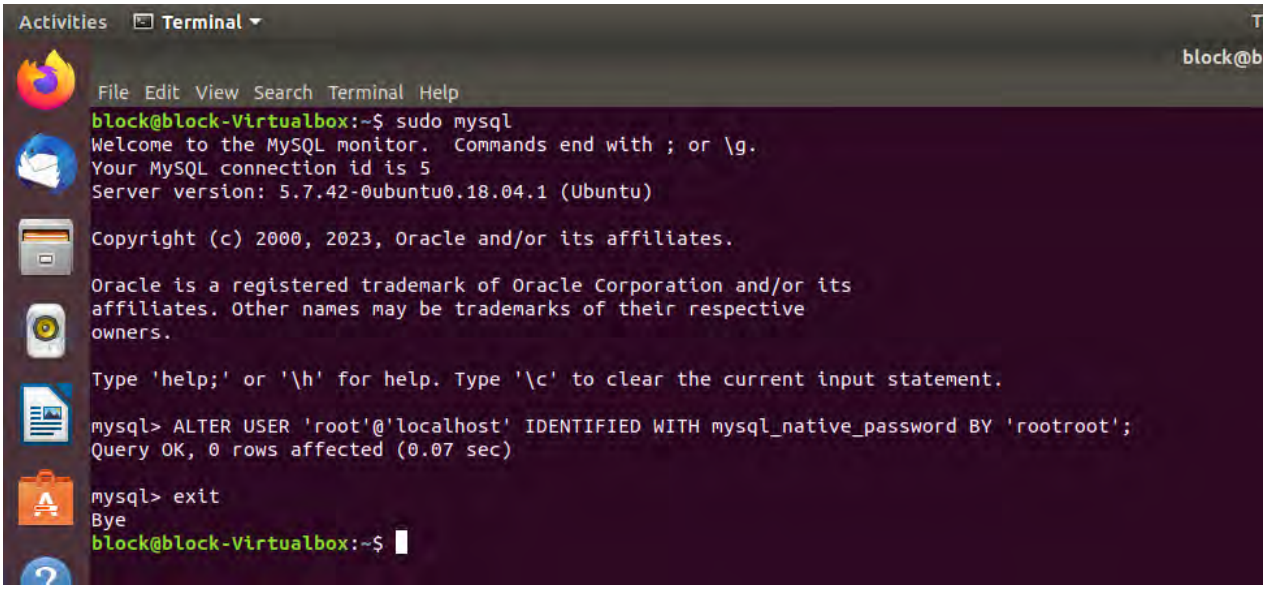

Figura 29: Estableciendo una contraseña

4.6.3 Creación de Usuarios y Base de Datos

Por medio de mysql se va a crear un usuario y una nueva base de datos. Se ejecuta el siguiente comando:

> *mysql –u root –p*

Tiene que ingresar la contraseña que puso en el paso anterior y va a agregar los siguientes datos:

> *mysql> CREATE DATABASE iota;*
>
> *mysql> CREATE USER ´iotauser´@´localhost´ IDENTIFIED BY ´root´;*
>
> *mysql> USE iota;*
>
> *mysql> GRANT ALL ON iota.* TO ´iotauser´@´localhost´;*
>
> *mysql> quit;*

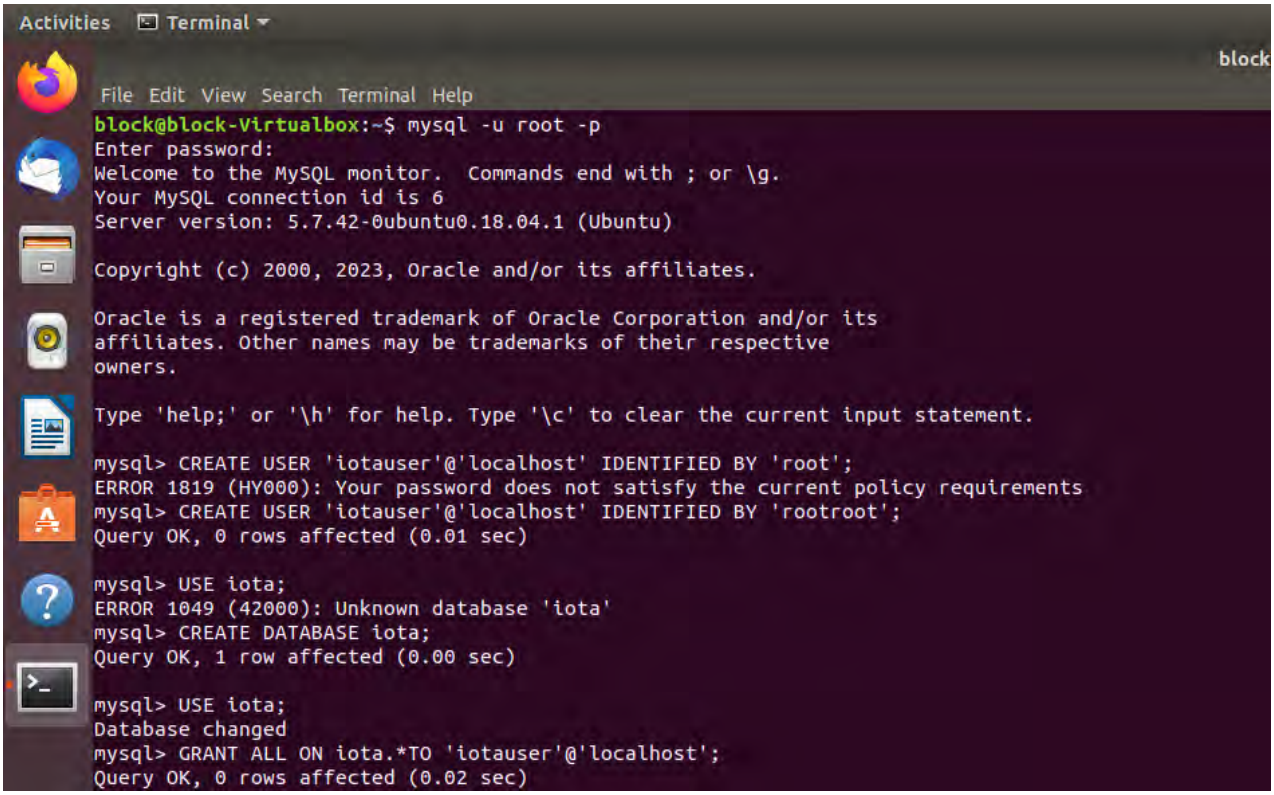

Figura 30: Creación de base de datos y Usuarios

4.6.4 Instalación de Phpmyadmin

Para una instalación correcto de PHPMyadmin y se evita el error de la siguiente imagen, es necesario seguir los pasos:

1. Instalar phpmyadmin. Comando a ejecutar: sudo apt-get install phpmyadmin

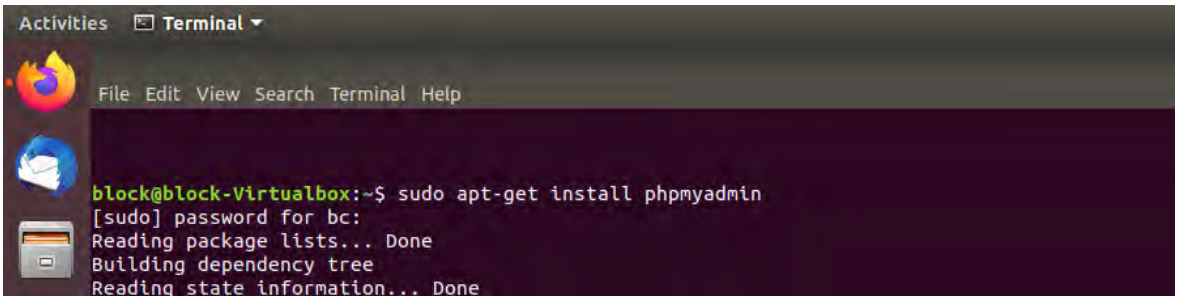

Figura 31: Código para instalar PHPMyAdmin

2. Reiniciar apache2, luego verificar el estado

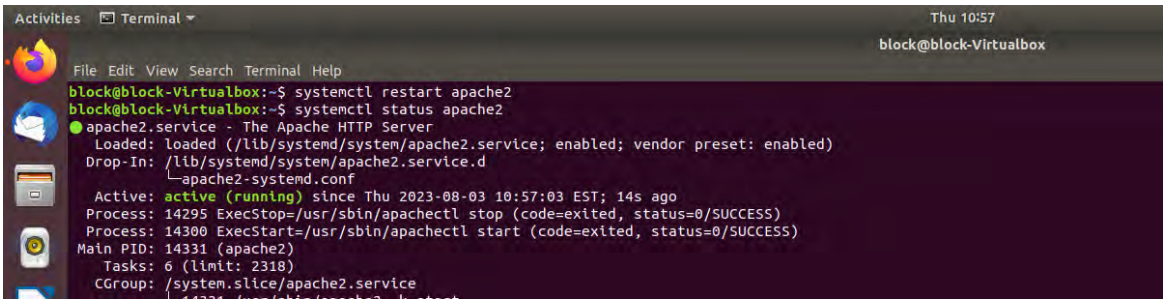

Figura 32: Estado del servidor Apache2

3. Entrar al archivo y agregar el siguiente comando:

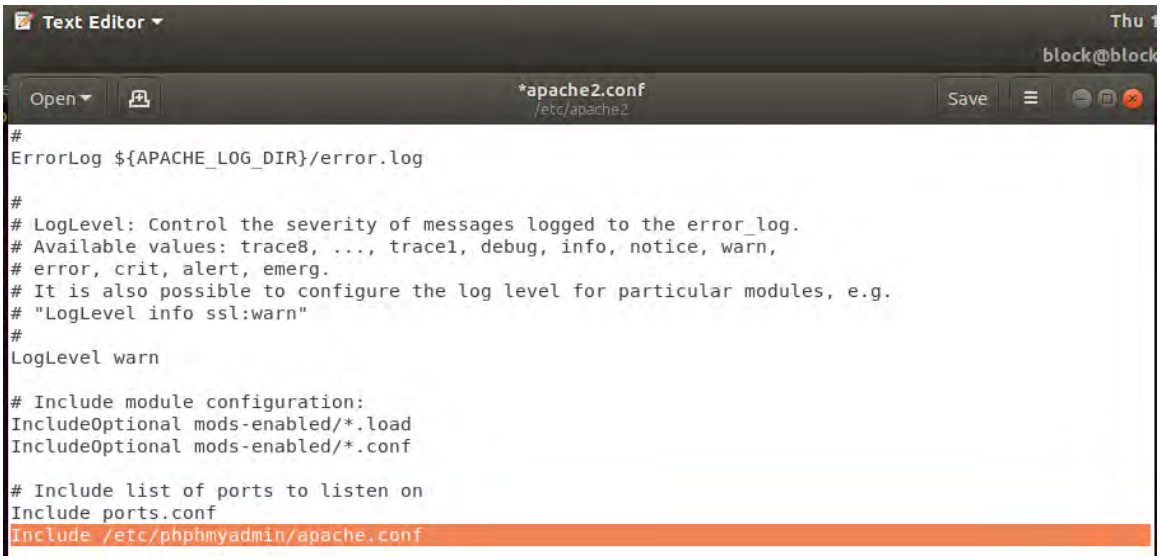

Figura 33: Modificación del archivo apache2.conf

4. Entrar con el siguiente link: http://localhost/phpmyadmin

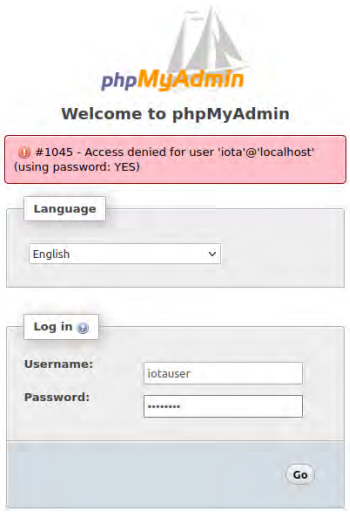

Figura 34: Inicio en phpMyAdmin

4.7 CÓDIGOS PARA EL RASPBERRY PI Y UBUNTU

Después de instalar y configurar los paquetes, controladores y dependencias necesarios. Luego procedemos a enviar los datos de los sensores a la Raspberry PI y luego a la máquina Ubuntu.

4.7.1 Lado Raspberry Pi

En el Raspberry Pi solo es necesario tener un archivo en Python cual va a enviar los datos a Ubuntu mediante MQTT. De primero se va a importar las librerías: *paho mqtt client, time, json y randint.*

```
import paho.mqtt.client as mqtt
import time
import json
from random import randint
```

Luego se tiene que establecer la IP del bróker:

```
mqtt_broker = "192.168.1.84"  # MQTT broker IP Address
pub_topic = "test/data"  # send messages to this topic
```

Se a completa el código con los datos que se requiere enviar, y para tener un mejor control se imprime los datos junto con el tiempo que se envió. Los datos se envían cada 8 segundos.

```
while True:
    coordinates = get_random_coordinates()
    timestamp = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
    client.publish(pub_topic, json.dumps(coordinates))
    print("Published data at", timestamp, ":", coordinates)
    time.sleep(8)
```

```
def get_random_coordinates():
    return {
        "x": randint(1, 10),
        "y": randint(1, 10),
        "z": randint(1, 10)
    }

client = mqtt.Client()
client.connect(mqtt_broker, 1883, 60)
```

4.7.2 Lado Ubuntu

En una carpeta se estarán los códigos ‘all.py’, ‘tobdb.py’, y ‘todatabase.py’ para almacenar los datos que el Raspberry Pi envíe. El archivo llamado ‘all.py’ va a compilar los archivos ‘‘tobdb.py’, y ‘todatabase.py’.

Código de ‘all.py’

```
import paho.mqtt.client as mqtt
import json
from tomysql import store_mysql
from tobdb import send_to_bdb  # Import the send_to_bdb function from tobdb
#import requests
import time

### MQTT SETTINGS ###
mqtt_broker = "172.16.0.26"  # MQTT broker IP Address

sub_topic = "test/data"  # receive messages on this topic

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe(sub_topic)

def on_message(client, userdata, msg):
    print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))

    sensor_data = json.loads(msg.payload)
```

```
    # Store data to MariaDB client
    store_mysql(sensor_data)

    # Send data to BigchainDB
    send_to_bdb(sensor_data)

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect(mqtt_broker, 1883, 60)

### Start the MQTT forever loop
client.loop_forever()

with open('datos.txt', 'wb') as dec_file:
    dec_file.write(decrypted)
```

4.7.3 Código para BigchainDB

```
from bigchaindb_driver import BigchainDB
import time

bdb_root_url = 'http://localhost:9984'
bdb = BigchainDB(bdb_root_url)

victor_priv = '6ymERLYuxSx7JWciFDDk6DwHDSt4pZYXdJLVAK7WkYAp'
victor_pub = 'AGbQL27WZSxz7cckL2nHtzbYtjjDRjkYXGqcGH8cBNT3'

asset_data = {
    'data': {
        'device': {
            'serial_number': 'JXDMD72',
            'owner': 'owner'
        },
    },
}

def send_to_bdb(sensor_data):
    asset_data['data']['device']['Storage TimeStamp'] = str(time.strftime('%Y-%m-
%d %H:%M:%S', time.localtime()))
```

4.7.4 Código para la base de datos

```
import time
import MySQLdb as mdb

### MYSQL SETTINGS ###

data_base_hostname = "localhost"      # MySQL host ip address or name
data_base_database = "iota"      # MySQL database name
data_base_username = "iotauser"      # MySQL database user name
data_base_password = "rootroot"      # MySQL database password

### MYSQL SETUP ###

data_base = mdb.connect(data_base_hostname, data_base_username,
data_base_password, data_base_database)
data_base_cursor = data_base.cursor()        # prepare a cursor object

def store_mysql(coordinates):
    vars_to_sql = []
    keys_to_sql = []

    for key, value in coordinates.items():
        if key == 'Data_Collection_Time_Stamp':
            value = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())

        vars_to_sql.append(value)
        keys_to_sql.append(key)

    keys_to_sql = ', '.join(keys_to_sql)
    placeholders = ', '.join(['%s'] * len(vars_to_sql))

    try:
        # Execute the SQL command
        queryText = f"INSERT INTO Environment_Parameters ({keys_to_sql}) VALUES
({placeholders})"
        data_base_cursor.execute(queryText, vars_to_sql)
        print('Successfully Added record to MySQL')
        data_base.commit()

    except mdb.Error as e:
        try:
            print("MySQL Error [%d]: %s" % (e.args[0], e.args[1]))
        except IndexError:
            print("MySQL Error: %s" % str(e))
        # Rollback in case there is any error
        data_base.rollback()
        print('ERROR adding record to MYSQL')
```

Capítulo V

5. ANÁLISIS DE RESULTADOS, INSERCIÓN DE DATOS EN CADENAS DE BLOQUES

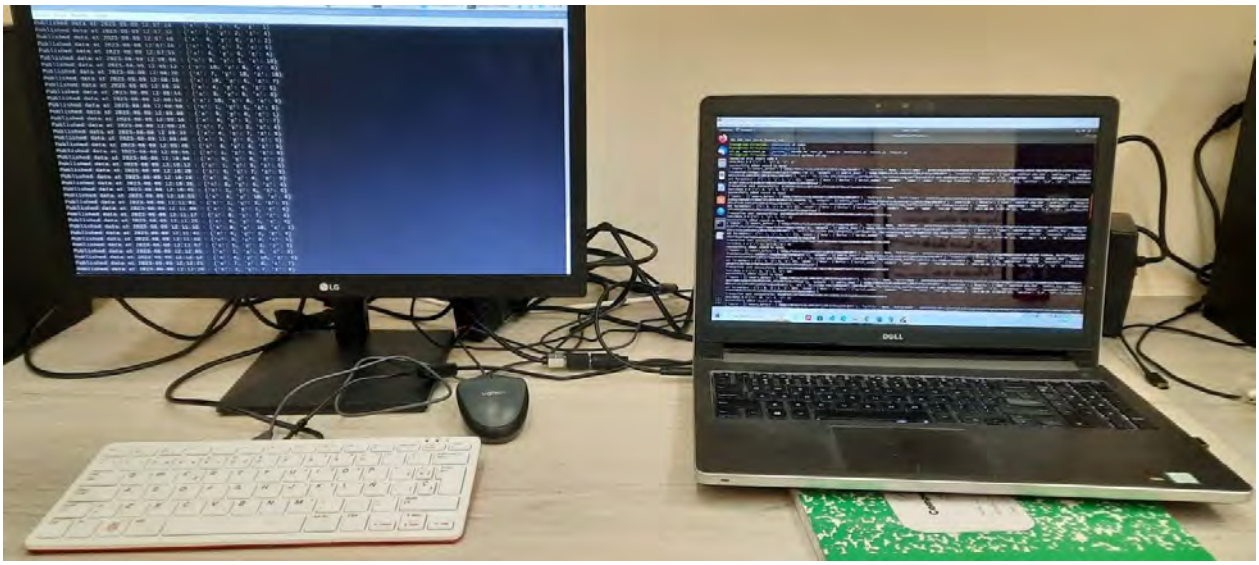

Figura 35: El Raspberry Pi (Lado Izquierdo) y La Máquina Ubuntu (Lado Derecho)

Se procede a ejecutar el programa del Raspberry para enviar datos, y al mismo tiempo el programa “all.py” en la maquina Ubuntu. A continuación, se puede visualizar los datos enviados por el Raspberry Pi:

```
Published data at 2023-08-09 00:32:52 : {'x': 9, 'y': 6, 'z': 1}
Published data at 2023-08-09 00:33:00 : {'x': 7, 'y': 8, 'z': 4}
Published data at 2023-08-09 00:33:08 : {'x': 10, 'y': 2, 'z': 5}
Published data at 2023-08-09 00:33:16 : {'x': 10, 'y': 2, 'z': 5}
Published data at 2023-08-09 00:33:24 : {'x': 4, 'y': 5, 'z': 9}
Published data at 2023-08-09 00:33:32 : {'x': 9, 'y': 2, 'z': 7}
Published data at 2023-08-09 00:33:40 : {'x': 8, 'y': 4, 'z': 1}
Published data at 2023-08-09 00:33:48 : {'x': 1, 'y': 6, 'z': 5}
Published data at 2023-08-09 00:33:56 : {'x': 9, 'y': 4, 'z': 2}
Published data at 2023-08-09 00:34:04 : {'x': 9, 'y': 9, 'z': 6}
Published data at 2023-08-09 00:34:12 : {'x': 3, 'y': 8, 'z': 5}
Published data at 2023-08-09 00:34:20 : {'x': 10, 'y': 9, 'z': 8}
```

Figura 36: Datos Enviados del Raspberry Pi a la Maquina Ubuntu

Queremos saber si los datos enviados son los correctos. La marca de tiempo nos ayudará a determinar si es correcta. Procedemos a resaltar cualquier línea, en cuyo caso, esta es: **x: 9, y:2, z:7.** Con tiempo: **2023-08-09 00:33:32.**

5.1 En lado Ubuntu

En la maquina se va a poder visualizar las coordenadas que recibe envidos por el Raspberry Pi. También se muestra que se agregó con éxito los datos a la base de datos y la transacción con su ID de bigchaindb se envió con éxito.

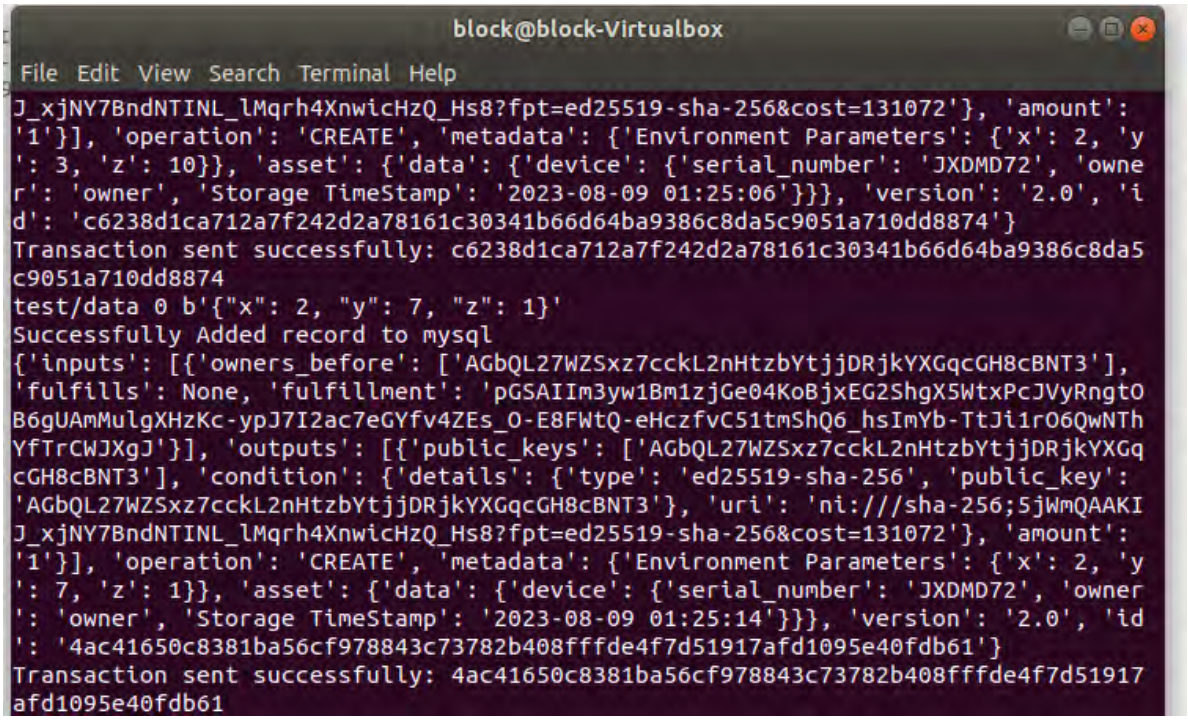

Figura 37: Coordenadas que recibe por el Raspberry Pi y la agregación con éxito de los datos la base de datos y la transacción de Blockchain

5.2 Base de Datos

Procedemos a verificar si se inserto correctamente en la base de datos mediante el comando http://localhost/phpmyadmin. Queremos buscar las coordenadas x,y,z con las siguientes características: **x: 9, y:2, z:7.** Con tiempo: **2023-08-09 00:33:32.**

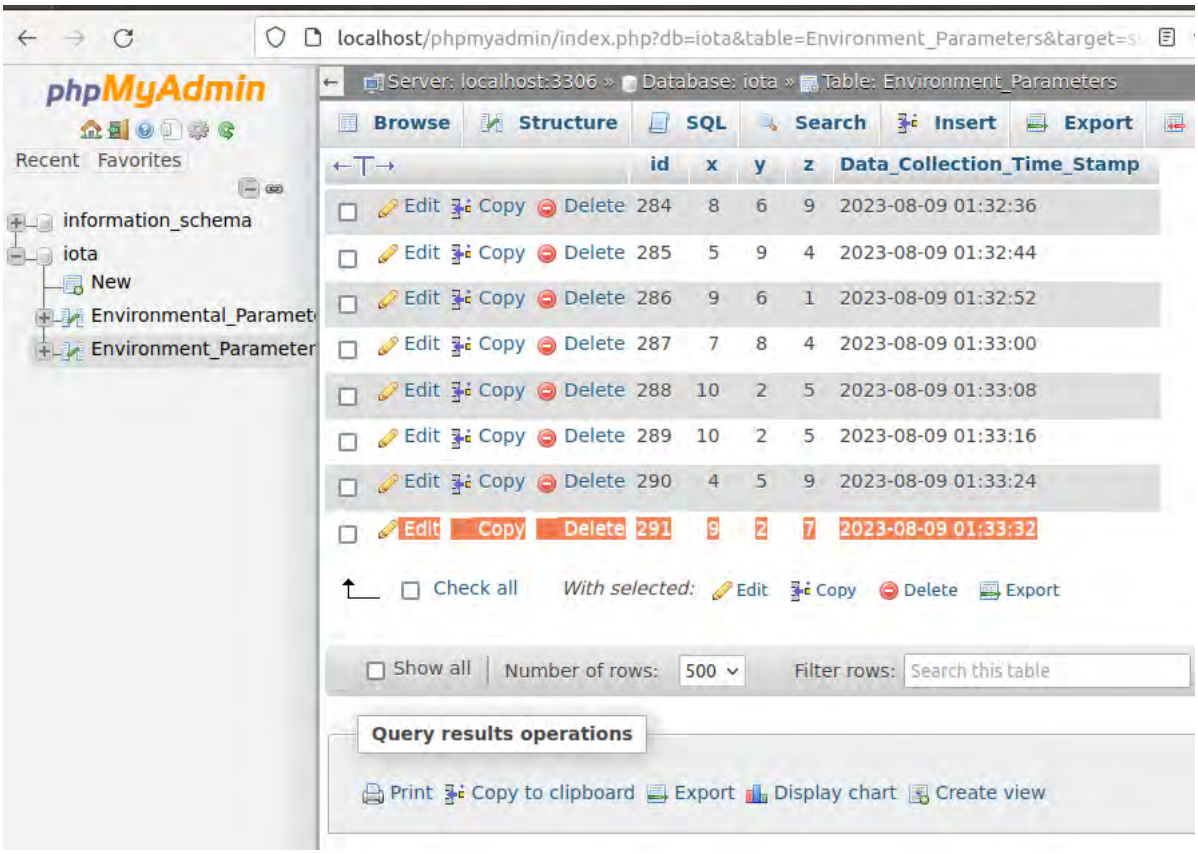

Figura 38: Base de Datos con coordenadas: x: 9, y:2, z:7. y tiempo, 2023-08-09 00:33:32.

El texto resaltado en naranja muestra que las coordenadas se guardaron en la base de datos.

5.3 BIGCHAINDB

De los resultados de la figura 39 se puede obtener la ID de transacción para visualir los datos en una pagina de navegador mediante el siguiente link: http://localhost:9984/api/v1/transactions/ID

En ID se debe de proporcionar su ID. En mi caso, se desea dar seguimiento a la coordenadas, : x: 9, y:2, z:7, con tiempo: 2023-08-09 00:33:32.

En la imagen se puede visualizar que efectivamente se agrego los datos a bigchaindb.

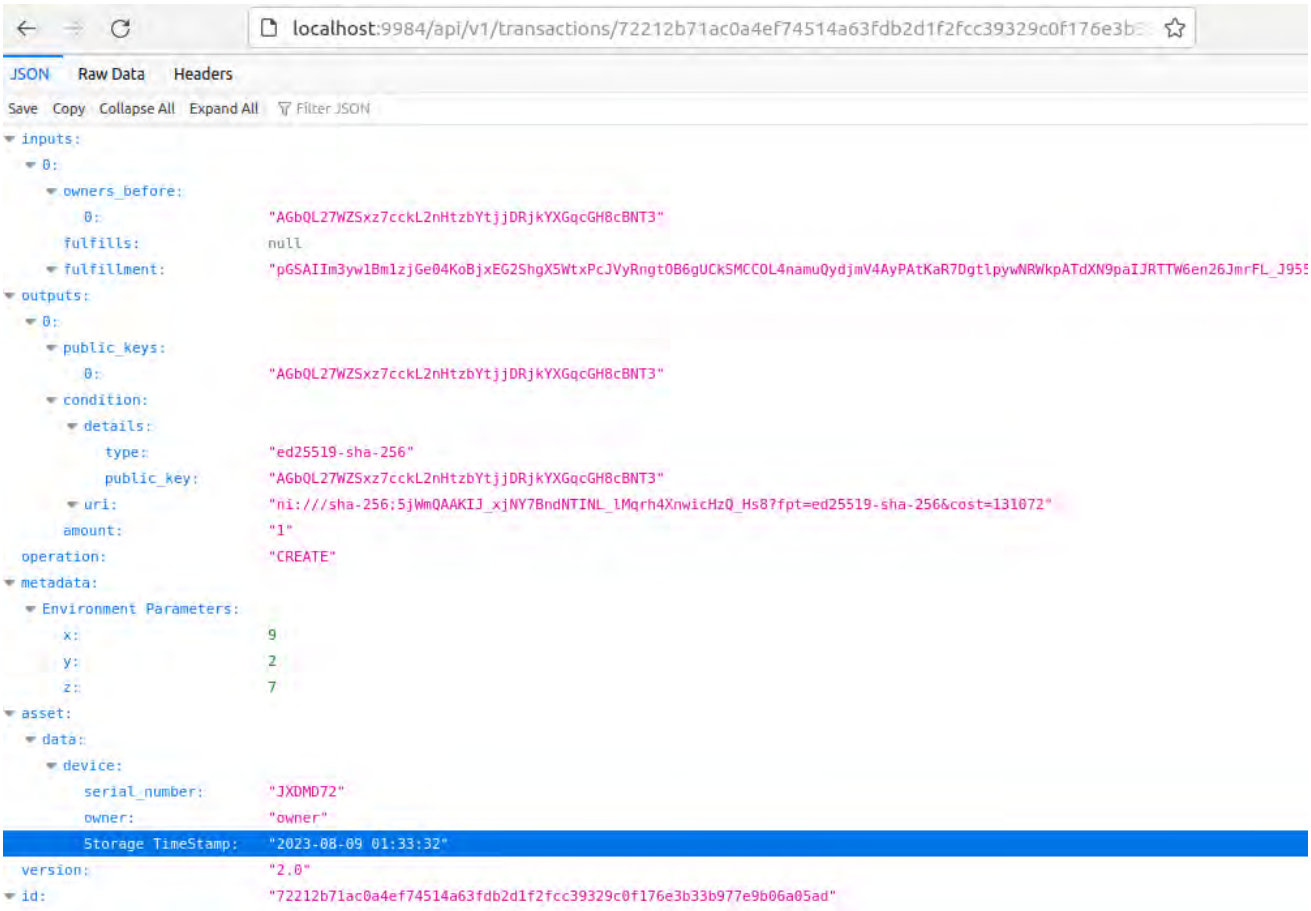

Figura 39: Transacción de las coordenadas

CONCLUSIONES

Los sistemas de posicionamiento en interiores utilizan sensores y tecnologías de comunicación para ubicar objetos en ambientes interiores. Los sistemas IPS pueden ser vulnerables a las amenazas en la seguridad de la información transmitida por los sensores, por tanto, requieren de autenticación, protección de datos, estabilidad, resistencia a los ataques, facilidad de implementación, auto mantenimiento, entre otras características. Las características de seguridad pueden ser aportadas por blockchain. En este proyecto se aplica la tecnología blockchain en un sistema de posicionamiento de objetos en espacios interiores.

GLOSARIO

Término	Definición
IPS (Indoor Positioning System)	Sistema que permite la detección de objetos en el interior de edificios donde normalmente no funcionan las estaciones de satélite.
RSSI (Received Signal Strength Indicator)	Mide que tan bien un dispositivo puede recibir una una señal.
Distributed Ledger (Base de Datos Distribuida)	No hay un administrador central y es una base de datos distribuida y mantenida por cada nodo en una red.
Blockchain (Cadena de bloques)	Sistema descentralizado para registrar transacciones y gestionar datos
Hash	Preserva la autenticidad de los datos en blockchain, es como una huella digital
P2P (Peer to Peer)	Sistemas que permiten la interacción directamente de individuos y trabajan como una organización colectiva.
Protocolo	Es un conjunto de reglas predefinidas para la estandarización y el intercambio de actividades informáticas.
Ethereum	Plataforma de blockchain descentralizada P2P que ejecuta y verifica de códigos de la aplicación, llamados contratos inteligentes.
JSON (JavaScript Objetc Notation)	Formato estándar basado en texto para compartir datos que utiliza texto legible para almacenar y transmitir datos, también se utiliza para representar datos estructurados basados en la sintaxis de objetos JavaScript.

BIBLIOGRAFIA

Ali, M. S., Vecchio, M., Pincheira, M., Dolui, K., Antonelli, F., & Rehmani, M. H. (2019). Applications of Blockchains in the Internet of Things: A Comprehensive Survey. *IEEE Communications Surveys & Tutorials, 21*(2), 1676-1717. doi:10.1109/COMST.2018.2886932

Alzoubi, Y. I., Al-Ahmad, A., Kahtan, H., & Jaradat, A. (2022). Internet of Things and Blockchain Integration: Security, Privacy, Technical, and Design Challenges. *futute internet, 14*(7), 216. Retrieved from https://doi.org/10.3390/fi14070216

Aragay, V. R. (2018). *Inserción de Datos de Sensores en Cadenas de Bloques Empleando Raspberry Pi y Tecnologías Blockchain.* Unibersidad Politécnica de Madrid . Madrid: Industriales ETSII UPM. Obtenido de https://oa.upm.es/53622/1/TFG_VICTOR_ROMAN_ARAGAY.pdf

Bardwell, J., & Akin, D. (2005). *Cetified Wireless Network Administrator.* McGraw-Hill.

Bashir, I. (2018). *Mastering Blockchain* (2nd ed.). (B. Rai, Ed.) Birmingham, United Kingdom: Packt Publishing Ltd.

BigchainDB GmbH. (14 de Mayo de 2018). BigchainDB 2.0 The Blockchain Database. *BigchainDB 2.0 Whitepaper, 1*(1), 1-14. Obtenido de https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf

Boneh, D., & Shoup, V. (2023). *A Graduate Course in Applied Cryptography* (6 ed.). Palo Alto: Springe.

Caramés, T. M., & Lamass, P. F. (2018, April 11). A Review on the Use of Blockchain for the Internet of Things. *IEEE Access, 6*, 32979-33001. doi:10.1109/ACCESS.2018.2842685

Carpena, M. G. (2018). *Aplicación de la tecnología blockchain a soluciones de la Internet de las Cosas (On blockchain and its usage in Internet of the Things networks).* UNIVERSIDAD DE CANTABRIA, Ingeniería de Tecnologías de Telecomunicación. Cantabria: E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION. Obtenido de http://repositorio.unican.es:8080/xmlui/handle/10902/14915

Conteron, O. D. (2021). *Sistema De Smart Home Aplicando IOT Y Blockchain .* UNIVERSIDAD TÉCNICA DE AMBATO, Electrónica: Tecnología de la Información y Sistemas de Control. Amabto - Ecuador: Carlos Gallegos. Obtenido de https://repositorio.uta.edu.ec/jspui/handle/123456789/33158

Koyuncu, H., & Yang, S. H. (2010). Survey of Indoor Positioning and Object Locating Systems. *International Journal of Computer Science and Network Security (IJCSNS), 10*(5).

Krichen, M., Ammi, M., Mihoub, A., & Almutiq, M. (2022). Blockchain for Modern Applications: A Survey. *Sensors, 22*(14), 5274. doi:10.3390/s22145274

Kumar, R., Khan, F., Kadry, S., & Rho, S. (19 de Noviembre de 2021). A Survey on blockchain for industrial Internet of Things. *Alexandria Engineering Journal , 61*(8), 6001-6022. doi:https://doi.org/10.1016/j.aej.2021.11.023

Kumar, V., & Ramesh, C. (Mayo de 2019). Storing IOT Data Securely in a Private Ethereum Blockchain. *UNLV*, 3582. Obtenido de http://dx.doi.org/10.34917/15778410

Mahtab, H. A. (2009). *Positioning System, Location Fingerprint, Wi-Fi, Bluetooth, User Feedback, Signal Strength Difference.* Sigapore: ScholarBank@NUS Repository. Retrieved from http://scholarbank.nus.edu.sg/handle/10635/23718

Mautz, R. (2012). *Indoor Positioning Technologies.* Institute of Geodesy and Photogrammetry, Department of Civil, Environmental and Geomatic Engineering. ETH Zurich Research Collection.

Niclas, K., Lammel, P., & Tcholtchev, N. (9 de Agosto de 2020). Prototype Implementation and Evaluation of a Blockchain Component on IoT Devices. *Procedia Computer Science, 175*, 379-386. Obtenido de https://doi.org/10.1016/j.procs.2020.07.054

Panarello, A., Tapas, N., Merlino, G., Longo, F., & Puliafito, A. (26 de Agosto de 2018). Blockchain and IoT Integration: A Systematic Survey . *Sensors, 18*(8), 2575. doi:10.3390/s18082575

Salman, T., Zolanvari, M., Erbad, A., Jain, R., & Samaka, M. (2019). Security Services Using Blockchains: A State of the Art Survey. *IEEE Communications Surveys & Tutorials , 1*(1), 99. doi:10.1109/COMST.2018.2863956

Seunghyeon, L., Seok, H.-W., Lee, K.-r., & In, H. P. (2022). B-GPS: Blockchain-Based Global Positioning System for Improved Data Integrity and Reliability. *International Journal of Geo-Information, 11*(3), 186. Obtenido de https://doi.org/10.3390/ijgi11030186

ANEXO

ANEXO A

Código para generar llaves en Python

```python
from bigchaindb_driver.crypto import generate_keypair

public_key, private_key = generate_keypair()

print(f"Public Key: {public_key}")
print(f"Private Key: {private_key}")
```

ANEXO B

Código del Archivo YML con BigchainDB y MongoDB

```yaml
version: '3'

services:
  mongo:
    image: mongo:latest
    ports:
      - "27017:27017" # MongoDB default port
    volumes:
      - ./data/mongodb:/data/db

  bigchaindb:
    image: bigchaindb/bigchaindb:latest
    ports:
      - "9984:9984" # API endpoint port
      - "9985:9985" # WebSocket port
    volumes:
      - ./data:/data
    environment:
      BIGCHAINDB_DATABASE_BACKEND: localmongodb
```

```
      BIGCHAINDB_DATABASE_HOST: mongodb://mongo/bigchain
      BIGCHAINDB_DATABASE_NAME: bigchain
      BIGCHAINDB_SERVER_FRONTEND_BIND: "172.16.0.34:9984"
      BIGCHAINDB_KEYPAIR_PUBLIC: "<FtR4WgDNJtz9jdqufix6PCVYcsjnmZJRAd5UEEhf48jf>"
      BIGCHAINDB_KEYPAIR_PRIVATE:
"<H5ooBQwYE3VoSQ8UDazWaW9GT7nTt4dyeiW88sQyuH39>"

#Public Key: FtR4WgDNJtz9jdqufix6PCVYcsjnmZJRAd5UEEhf48jf
#Private Key: H5ooBQwYE3VoSQ8UDazWaW9GT7nTt4dyeiW88sQyuH39
```

ANEXO C

Código del archivo YML para Tendermint, Bigchaindb y MongoDB

```
version: '3'
services:
  bigchaindb:
    image: bigchaindb/bigchaindb
    volumes:
      - bigchaindb_data:/data/db
      - ./config.json:/etc/bigchaindb/config.json
    ports:
      - "9984:9984"
      - "9985:9985"
    environment:
      BIGCHAINDB_DATABASE_HOST: mongodb
      BIGCHAINDB_DATABASE_BACKEND: localmongodb
    networks:
      - bigchaindb_net

  mongodb:
    image: mongo:latest
    volumes:
      - mongodb_data:/data/db
    networks:
      - bigchaindb_net

  tendermint:
    image: tendermint/tendermint:v0.34.9
    ports:
      - "26656:26656"
```

```
      - "26657:26657"
    volumes:
      - tendermint_data:/tendermint
    networks:
      - bigchaindb_net

volumes:
  bigchaindb_data:
    driver: local
  mongodb_data:

  tendermint_data:

networks:
  bigchaindb_net:
    driver: bridge
```

ANEXO D

Código para enviar datos y establecer la conexión MQTT en el Raspberry Pi

```
import paho.mqtt.client as mqtt
import time
import json
from random import randint

mqtt_broker = "192.168.1.84"  # MQTT broker IP Address

pub_topic = "test/data"  # send messages to this topic

def get_random_coordinates():
    return {
        "x": randint(1, 10),
        "y": randint(1, 10),
        "z": randint(1, 10)
    }

client = mqtt.Client()
client.connect(mqtt_broker, 1883, 60)

while True:
    coordinates = get_random_coordinates()
```

```
    timestamp = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
    client.publish(pub_topic, json.dumps(coordinates))
    print("Published data at", timestamp, ":", coordinates)
    time.sleep(8)
```

ANEXO E

Código para la maquina Ubuntu y Correr el código de la base de datos y bigchaindb.

```
import paho.mqtt.client as mqtt
import json
from tomysql import store_mysql
from tobdb import send_to_bdb  # Import the send_to_bdb function from tobdb
#import requests
import time

### MQTT SETTINGS ###
mqtt_broker = "172.16.0.26"  # MQTT broker IP Address

sub_topic = "test/data"  # receive messages on this topic

def on_connect(client, userdata, flags, rc):
    print("Connected with result code " + str(rc))
    client.subscribe(sub_topic)

def on_message(client, userdata, msg):
    print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))

    sensor_data = json.loads(msg.payload)

    # Store data to MariaDB client
    store_mysql(sensor_data)

    # Send data to BigchainDB
    send_to_bdb(sensor_data)

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect(mqtt_broker, 1883, 60)

### Start the MQTT forever loop
client.loop_forever()
```

ANEXO F

Código para la base de datos

```
import time
import MySQLdb as mdb

### MYSQL SETTINGS ###

data_base_hostname = "localhost"      # MySQL host ip address or name
data_base_database = "iota"      # MySQL database name
data_base_username = "iotauser"      # MySQL database user name
data_base_password = "rootroot"      # MySQL database password

### MYSQL SETUP ###

data_base = mdb.connect(data_base_hostname, data_base_username,
data_base_password, data_base_database)
data_base_cursor = data_base.cursor()        # prepare a cursor object

def store_mysql(coordinates):
    vars_to_sql = []
    keys_to_sql = []

    for key, value in coordinates.items():
        if key == 'Data_Collection_Time_Stamp':
            value = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())

        vars_to_sql.append(value)
        keys_to_sql.append(key)

    keys_to_sql = ', '.join(keys_to_sql)
    placeholders = ', '.join(['%s'] * len(vars_to_sql))

    try:
        # Execute the SQL command
        queryText = f"INSERT INTO Environment_Parameters ({keys_to_sql}) VALUES
({placeholders})"
        data_base_cursor.execute(queryText, vars_to_sql)
        print('Successfully Added record to MySQL')
        data_base.commit()

    except mdb.Error as e:
        try:
```

```
            print("MySQL Error [%d]: %s" % (e.args[0], e.args[1]))
        except IndexError:
            print("MySQL Error: %s" % str(e))
        # Rollback in case there is any error
        data_base.rollback()
        print('ERROR adding record to MYSQL')
```

ANEXO G

Código para bigchaindb

```
from bigchaindb_driver import BigchainDB
import time

bdb_root_url = 'http://localhost:9984'
bdb = BigchainDB(bdb_root_url)

victor_priv = '6ymERLYuxSx7JWciFDDk6DwHDSt4pZYXdJLVAK7WkYAp'
victor_pub = 'AGbQL27WZSxz7cckL2nHtzbYtjjDRjkYXGqcGH8cBNT3'

asset_data = {
    'data': {
        'device': {
            'serial_number': 'JXDMD72',
            'owner': 'owner'
        },
    },
}

def send_to_bdb(sensor_data):
    asset_data['data']['device']['Storage TimeStamp'] = str(time.strftime('%Y-%m-
%d %H:%M:%S', time.localtime()))
    metadata = {'Environment Parameters': sensor_data}

    prepared_creation_tx = bdb.transactions.prepare(
        operation='CREATE',
        signers=victor_pub,
        asset=asset_data,
        metadata=metadata
    )
```

```
fulfilled_creation_tx = bdb.transactions.fulfill(
    prepared_creation_tx,
    private_keys=victor_priv
)

print(fulfilled_creation_tx)

try:
    sent_creation_tx = bdb.transactions.send_commit(fulfilled_creation_tx)
    print("Transaction sent successfully:", sent_creation_tx['id'])
except Exception as e:
    print("Error sending transaction:", e)
```