ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Diseño y prototipo de un sistema de control de acceso a puertas aplicando blockchain con IoT mediante smart contracts.

EXAMEN COMPLEXIVO

Previo la obtención del Título de:

Magíster en Telecomunicaciones

Presentado por:
Iván Olmedo Armijo Guamán

GUAYAQUIL - ECUADOR
Año: 2025

DEDICATORIA

A mi familia, por su amor incondicional, su apoyo inquebrantable y por ser mi mayor inspiración en cada paso de este camino. Este logro también es suyo.

AGRADECIMIENTOS

Primero que todo, quiero expresar mi gratitud a Dios por todas las bendiciones que nos brinda diariamente y por permitirnos seguir adelante en este mundo. En segundo lugar, quiero agradecer a mi familia, que ha estado a mi lado en los momentos buenos y difíciles, y en especial a mi novia, cuyo apoyo incondicional ha sido fundamental en todo momento.

DECLARACIÓN EXPRESA

Yo Iván Olmedo Armijo Guamán acuerdo y reconozco que: La titularidad de los derechos patrimoniales de autor (derechos de autor) del proyecto de graduación corresponderá al autor o autores, sin perjuicio de lo cual la ESPOL recibe en este acto una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra con facultad de sublicenciar, incluyendo la autorización para su divulgación, así como para la creación y uso de obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores. El o los estudiantes deberán procurar en cualquier caso de cesión de sus derechos patrimoniales incluir una cláusula en la cesión que proteja la vigencia de la licencia aquí concedida a la ESPOL.

La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, secreto empresarial, derechos patrimoniales de autor sobre software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada por mí/nosotros durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que me/nos corresponda de los beneficios económicos que la ESPOL reciba por la explotación de mi/nuestra innovación, de ser el caso.

En los casos donde la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique al/los autor/es que existe una innovación potencialmente patentable sobre los resultados del proyecto de graduación, no se realizará publicación o divulgación alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 4 de febrero del 2025.

Firmado electrónicamente por:
IVAN OLMEDO ARMIJO GUAMAN

Iván Olmedo Armijo Guamán

EVALUADORES

Firmado electrónicamente por:
MARIA ANTONIETA
ALVAREZ VILLANUEVA

María Antonieta Álvarez, PhD

PROFESOR EVALUADOR

Firmado electrónicamente por:
RICARDO ALFREDO
CAJO DIAZ

Ricardo Cajo, PhD

PROFESOR EVALUADOR

RESUMEN

Este proyecto plantea la implementación de un sistema de control de acceso fundamentado en Blockchain, IoT y ThingsBoard Cloud, con el objetivo de mejorar la seguridad, trazabilidad y automatización en la gestión de accesos. La solución permite validar credenciales mediante contratos inteligentes en una blockchain privada, controlar la apertura de puertas con Raspberry Pi y relés, y registrar eventos en la nube para su monitoreo en tiempo real. Se justifica su desarrollo por la necesidad de un sistema confiable y descentralizado que evite manipulaciones en los registros y garantice la autenticación segura de los usuarios.

Para la implementación, se utilizó Ganache como blockchain privada, Node.js para gestionar la comunicación entre la interfaz web, la blockchain y thingsboard cloud para la visualización de eventos. Se instalaron y configuraron las herramientas necesarias en Windows y Raspberry Pi, y se desarrollaron scripts para la interacción con los dispositivos IoT. Se realizaron pruebas de validación de accesos, apertura de puertas y notificaciones mediante MQTT y Nodemailer.

Los resultados evidenciaron que el sistema validó los accesos en menos de un segundo, asegurando la integridad de los registros y la eficiencia en la automatización del proceso. La integración con ThingsBoard permitió visualizar eventos en tiempo real, aunque se detectaron retrasos ocasionales por la dependencia de la conexión a internet.

Se concluye que la solución es eficiente y escalable, permitiendo su implementación en entornos donde la seguridad y la trazabilidad son críticas. Se recomienda optimizar la infraestructura de red y evaluar la migración a una blockchain pública para mayor confiabilidad.

Palabras Clave: Blockchain, IoT, ThingsBoard, Seguridad, Acceso Inteligente.

ABSTRACT

This project proposes the implementation of an access control system based on Blockchain, IoT and ThingsBoard Cloud, with the aim of improving security, traceability and automation in access management. The solution allows validating credentials through smart contracts in a private blockchain, controlling the opening of doors with Raspberry Pi and relays, and recording events in the cloud for real-time monitoring. Its development is justified by the need for a reliable and decentralized system that prevents manipulation of records and guarantees secure user authentication.

For the implementation, Ganache was used as a private blockchain, Node.js to manage communication between the web interface, the blockchain and thingsboard cloud for event visualization. The necessary tools were installed and configured on Windows and Raspberry Pi, and scripts were developed for interaction with IoT devices. Access validation, door opening and notification tests were carried out using MQTT and Nodemailer.

The results showed that the system validated the accesses in less than a second, ensuring the integrity of the records and the efficiency of the automation of the process. The integration with ThingsBoard allowed viewing events in real time, although occasional delays were detected due to the dependence on the internet connection.

It is concluded that the solution is efficient and scalable, allowing its implementation in environments where security and traceability are critical. It is recommended to optimize the network infrastructure and evaluate the migration to a public blockchain for greater reliability.

Keywords: Blockchain, IoT, ThingsBoard, Security, Smart Access.

ÍNDICE GENERAL

ABREVIATURAS

IOT	Internet of Things
MQQT	Message Queuing Telemetry Transport
EVM	Ethereum Virtual Machine
GPIO	General Purpose Input/Output
API	Interfaz de Programación de Aplicaciones
ABI	Application Binary Interface
URL	Uniform Resource Locator
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
VCC	voltaje de corriente continua

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad, los sistemas de control de acceso han experimentado importantes avances, motivados por la creciente demanda de proteger tanto entornos físicos como digitales. Una de las innovaciones tecnológicas que está transformando este ámbito es la integración de blockchain y IoT (Internet de las Cosas), con el uso destacado de los smart contracts o contratos inteligentes. Estas tecnologías no solo fortalecen la seguridad, sino que también optimizan la eficiencia y promueven una mayor transparencia en la administración de accesos. [1].

El control de acceso es fundamental en diversos entornos, como oficinas, residencias y áreas industriales, donde se busca limitar y supervisar la entrada a ciertos espacios. No obstante, las soluciones tradicionales de control de acceso tienen algunas desventajas, como la centralización de datos, la exposición a ataques cibernéticos y la falta de transparencia en los registros. En este contexto, la tecnología blockchain se presenta como una solución innovadora [2].

La blockchain, al ser una tecnología descentralizada y distribuida, ofrece un enfoque seguro y confiable para la gestión de accesos mediante la creación de registros inmutables y visibles. Al combinarla con dispositivos IoT, que permiten el control remoto de cerraduras electrónicas, se puede desarrollar un sistema más inteligente y automatizado. Los smart contracts, que operan en la blockchain, son esenciales porque permiten ejecutar las reglas de acceso automáticamente, sin intermediarios, garantizando un sistema eficiente y seguro [3].

Se plantea la creación y desarrollo de un prototipo para gestionar el acceso a puertas mediante la integración de blockchain, IoT y contratos inteligentes, con el propósito de ofrecer una solución innovadora y avanzada para la seguridad en accesos físicos. Ésta tecnología tiene la capacidad de optimizar los procesos de verificación de identidad, aumentar la transparencia en los registros de acceso y ofrecer una mayor resistencia a posibles amenazas o fraudes [4].

CAPÍTULO 1

1. INTRODUCCIÓN

1.1 Descripción del problema

Los sistemas tradicionales de control de acceso presentan vulnerabilidades debido a su infraestructura centralizada, lo que los hace susceptibles a ciberataques, falta de transparencia y dificultades en la trazabilidad. Además, enfrentan limitaciones en escalabilidad al integrar múltiples dispositivos y usuarios, dependen de terceros para la administración de accesos, tienen procesos ineficientes para la asignación y revocación de permisos y generan altos costos operativos por su mantenimiento constante. La integración de blockchain e IoT ofrece una solución innovadora al proporcionar descentralización, trazabilidad mediante registros inmutables, transparencia en la gestión de accesos, automatización a través de smart contracts y mayor seguridad gracias a principios criptográficos, mejorando así la eficiencia y adaptabilidad en entornos dinámicos y conectados.

1.2 Justificación del problema

El acceso a infraestructuras críticas y áreas restringidas requiere soluciones seguras y eficientes que minimicen riesgos de intrusión, manipulación de datos y accesos no autorizados. Los métodos convencionales de control de acceso, que emplean llaves físicas, tarjetas RFID o contraseñas almacenadas en bases de datos centralizadas, son susceptibles a diversas vulnerabilidades, como extravío, clonación, falsificación de credenciales y alteraciones no autorizadas en los registros de acceso.

Este proyecto justifica su desarrollo en la necesidad de una solución descentralizada, segura y auditable que garantice la autenticidad de los accesos y la trazabilidad de eventos sin posibilidad de manipulación. La tecnología blockchain, al ser inmutable y distribuida, permite registrar cada acceso en un sistema transparente, evitando fraudes y accesos ilegales. Además, la integración con Internet de las Cosas (IoT) y la

plataforma ThingsBoard Cloud proporciona un monitoreo en tiempo real, mejorando la gestión y respuesta ante accesos sospechosos.

La automatización mediante contratos inteligentes en una blockchain privada (Ganache) y el control de relés con Raspberry Pi permite un sistema autónomo y seguro, donde solo los usuarios con permisos registrados pueden activar la apertura de puertas. A diferencia de los sistemas tradicionales, esta arquitectura ofrece seguridad avanzada, ya que cada acceso es verificado en la blockchain y registrado en la nube, garantizando auditoría, escalabilidad y confiabilidad

1.3 Objetivos

1.3.1 Objetivo General

- Diseñar un sistema de control de acceso a puertas aplicando blockchain con IoT mediante smart contracts.

1.3.2 Objetivos Específicos

- Desarrollar el código mediante el terminal de programación para la integración de Blockchain e IoT
- Implementar el sistema de control de acceso mediante un módulo Raspberry pi4 y blockchain para la apertura de puertas.
- Visualizar el registro de accesos en la nube a través de la plataforma de Internet de las cosas.

1.4 Propuesta de la solución

La solución propuesta consiste en la implementación de un sistema inteligente de control de acceso, utilizando un contrato inteligente desplegado en una red blockchain privada (Ganache). Este sistema permite autenticar usuarios y restringir el acceso a diferentes puertas según las credenciales y permisos asignados a cada usuario. El sistema combina elementos de hardware, como relés controlados mediante GPIO, y software, incluyendo una interfaz web para interactuar con el sistema, validación de

usuarios mediante el contrato inteligente y notificaciones por correo electrónico para registrar los eventos de acceso.

La solución busca abordar problemas comunes en sistemas de acceso tradicionales, como la falta de registro en tiempo real, restricciones de acceso configurables y notificaciones inmediatas. Además, la propuesta utiliza la transparencia y la seguridad de blockchain para evitar manipulaciones de datos y garantizar un registro inalterable de los eventos de acceso. Esto permite no solo un control eficiente, sino también una trazabilidad robusta de los accesos realizados.

En conjunto, la solución integra tecnologías modernas como blockchain, sistemas embebidos y servicios en la nube, ofreciendo una herramienta práctica, segura y escalable para el control de acceso en entornos diversos, como oficinas, edificios residenciales o industriales

1.5 Alcance y consideraciones

El alcance del proyecto incluye el desarrollo de un sistema de control de acceso inteligente que permita autenticar usuarios y restringir el acceso a puertas específicas en función de credenciales y permisos configurados en el contrato inteligente. Se integrará una red privada basada en Ganache para registrar y validar los accesos de manera segura, transparente e inalterable. También se incluye el envío de notificaciones de acceso en tiempo real mediante correos electrónicos al administrador o al usuario autorizado, además de una interfaz web intuitiva para que los usuarios ingresen sus credenciales y seleccionen la puerta deseada. Asimismo, el sistema registrará todos los eventos de acceso en la blockchain para garantizar trazabilidad y permitirá la escalabilidad hacia múltiples usuarios, puertas y futuras integraciones tecnológicas.

Las consideraciones del proyecto incluyen garantizar la seguridad de las contraseñas y datos de los usuarios mediante hashing o cifrado, además de implementar políticas de acceso restrictivo. También se contempla la dependencia del sistema respecto al hardware (relés y Raspberry Pi) y la compatibilidad con otros dispositivos que podrían requerir modificaciones. Al utilizar una red blockchain privada como Ganache, se reconoce que este entorno es limitado y que se requerirá una red más robusta para un

uso real. Se deberá prever la disponibilidad del servidor, la conectividad de la red y la sincronización del nodo blockchain, así como manejar escenarios de error como fallos en la validación de credenciales o desconexiones de red.

CAPÍTULO 2

2. MARCO TEORICO

2.1 Blockchain

La tecnología blockchain es un sistema distribuido de registro que permite almacenar información de manera segura, transparente e inmutable. Se compone de bloques enlazados entre sí que contienen datos y están protegidos por criptografía. Cada bloque incluye un hash del bloque anterior, lo que asegura la integridad del sistema. Esta tecnología fue introducida inicialmente con el lanzamiento de Bitcoin en 2008 y desde entonces ha sido adoptada en diferentes sectores, como finanzas, logística y gestión de identidades [5]. Blockchain se caracteriza por ser descentralizado, resistente a fallos y seguro contra manipulaciones.

2.2 Internet de las Cosas (IoT)

El Internet de las Cosas (IoT) es un modelo tecnológico que integra objetos físicos al mundo digital a través de sensores, actuadores y redes de comunicación. IoT permite la automatización y el monitoreo en tiempo real de dispositivos, lo que ha transformado industrias como la manufactura, el transporte y la domótica. Sin embargo, su implementación plantea desafíos en términos de seguridad y escalabilidad. Blockchain complementa a IoT proporcionando integridad de datos, eliminación de puntos únicos de falla y mejor control sobre los dispositivos conectados [6].

2.3 Ganache

Ganache es una plataforma de desarrollo para Ethereum que facilita la simulación de una blockchain local para los desarrolladores. Proporciona un entorno rápido y configurable para desplegar contratos inteligentes y probar aplicaciones descentralizadas. Ganache facilita la interacción con contratos inteligentes sin necesidad de utilizar redes públicas de blockchain, lo que reduce costos y tiempos de implementación en la etapa de desarrollo [7]. Su interfaz proporciona información detallada sobre transacciones, balances y contratos.

2.4 Ethereum

Ethereum es una red blockchain pública y descentralizada que posibilita la ejecución de contratos inteligentes. Además, incorpora una máquina virtual descentralizada, denominada Ethereum Virtual Machine (EVM), ejecuta scripts utilizando una red global de nodos también se la considera como una infraestructura para aplicaciones descentralizadas que automatizan procesos a través de contratos inteligentes [8].

2.5 Smart Contracts

Los contratos inteligentes son códigos programados en la blockchain que se activan de forma automática cuando se cumplen condiciones previamente establecidas. Estos eliminan la necesidad de intermediarios y garantizan la transparencia y seguridad de las operaciones. En el contexto de control de accesos, los contratos inteligentes pueden gestionar credenciales y registrar eventos en tiempo real, lo que asegura la trazabilidad y confiabilidad del sistema [9].

2.6 ThingsBoard

ThingsBoard es una solución de IoT de código abierto que permite capturar, procesar y mostrar datos de dispositivos conectados en tiempo real. Ofrece herramientas para la creación de dashboards interactivos y gestión de dispositivos IoT. En este proyecto, ThingsBoard se utiliza para visualizar eventos de acceso, monitorear el estado de puertas y analizar datos en tiempo real [10]. Su arquitectura modular y escalable la convierte en una solución ideal para aplicaciones IoT basadas en blockchain.

2.7 Node.js

Node.js es un entorno de ejecución para JavaScript diseñado para el desarrollo de aplicaciones del lado del servidor. Utiliza el motor V8 de Google Chrome y se destaca por su alto rendimiento y su capacidad para gestionar múltiples conexiones simultáneamente. Es ampliamente utilizado en el desarrollo de aplicaciones descentralizadas debido a su capacidad para integrar bibliotecas como Web3.js y manejar datos en tiempo real [11].

2.8 Web3.js

Web3.js es un conjunto de herramientas en JavaScript que permite la conexión y el acceso a la blockchain de Ethereum. Proporciona funcionalidades para conectar aplicaciones descentralizadas con contratos inteligentes, gestionar cuentas y realizar transacciones. Web3.js es esencial en este proyecto para ejecutar funciones de validación de acceso y registrar eventos en la blockchain desde una interfaz web [12].

2.9 Raspberry Pi 4

La Raspberry Pi 4 es un ordenador de placa única, asequible y de alto rendimiento. Cuenta con un procesador Broadcom BCM2711, arquitectura ARM Cortex-A72 de 64 bits, hasta 8 GB de RAM, conectividad Ethernet Gigabit**,** Wi-Fi de doble banda**,** Bluetooth 5.0, y múltiples interfaces de entrada y salida, como USB 3.0, HDMI y GPIO [13]. Su versatilidad la hace ideal para aplicaciones en Internet de las Cosas (IoT), automatización, aprendizaje de máquinas y entornos de desarrollo de servidores.

2.10 Módulo Relay para Raspberry Pi

El módulo relay permite controlar dispositivos eléctricos mediante señales de baja potencia generadas por la Raspberry Pi. Un relay es un interruptor electromecánico que se activa con una señal digital proveniente de los pines GPIO (General Purpose Input/Output) de la Raspberry. Estos módulos pueden manejar tensiones altas o bajas, permitiendo la automatización de circuitos eléctricos con seguridad y eficiencia. Se utilizan en sistemas de domótica, control de acceso y automatización industrial [14].

2.11 GPIO

Los pines GPIO de la Raspberry Pi permiten interactuar con dispositivos externos como sensores, actuadores y módulos de comunicación. La Raspberry Pi 4 cuenta con 40 pines GPIO, que pueden configurarse como entradas o salidas digitales. Para controlarlos, se pueden utilizar bibliotecas en Python (RPi.GPIO, gpiozero) o en JavaScript con Node.js (onoff, pigpio)**,** permitiendo una programación flexible y adaptada a proyectos de automatización y control de acceso [15].

2.12 Express

Express es un marco de trabajo liviano para aplicaciones web en Node.js, diseñado para desarrollar APIs y servidores con una arquitectura modular. Su uso es común en arquitecturas RESTful, facilitando la comunicación entre dispositivos IoT y plataformas web. En sistemas de control de acceso, Express.js es capaz de administrar solicitudes de acceso, procesar datos en tiempo real y manejar la autenticación de usuarios. [16].

2.13 Truffle

Truffle es una herramienta de desarrollo para Ethereum que permite desplegar y gestionar contratos inteligentes en la blockchain. Su uso en control de acceso basado en blockchain permite la ejecución de reglas automatizadas mediante contratos inteligentes, asegurando registros inmutables de accesos. Truffle facilita la creación, prueba e implementación de contratos en redes blockchain, optimizando la seguridad y la automatización de sistemas descentralizados [17]

2.14 Solidity

Solidity es un lenguaje de programación enfocado en contratos, diseñado para desarrollar smart contracts en plataformas blockchain compatibles con la Ethereum Virtual Machine. Fue desarrollado por Gavin Wood en 2014 y permite la ejecución de contratos autoejecutables sin intermediarios. Solidity es un lenguaje de tipado estático, influenciado por JavaScript, Python y C++, que se compila en bytecode para su ejecución en la EVM [18].

CAPÍTULO 3

3. DISEÑO E IMPLEMENTACIÓN

3.1 Diseño del sistema

El sistema se basa en una arquitectura modular con los siguientes componentes:

- **Blockchain privada (Ganache):** Almacena los eventos de acceso y gestiona los permisos de los usuarios mediante contratos inteligentes.
- **Servidor Node.js:** Actúa como intermediario entre la interfaz web, la blockchain, ThingsBoard y el hardware.
- **Interfaz web:** Permite a los usuarios ingresar credenciales y solicitar acceso a las puertas.
- **Raspberry Pi:** Controla los relés para activar/desactivar las cerraduras.
- **ThingsBoard:** Plataforma IoT utilizada para la visualización de eventos en tiempo real.
- **Nodemailer:** Facilita el envío de alertas por correo electrónico cada vez que un usuario ingresa al sistema.

El usuario solicita acceso desde la interfaz web, y el servidor Node.js verifica en la blockchain si tiene permisos. Si está autorizado, Node.js envía una señal a la Raspberry Pi para abrir la puerta. Al mismo tiempo, ThingsBoard registra el evento en tiempo real, y Nodemailer envía una notificación al usuario o administrador sobre el acceso (ver figura 3.1).

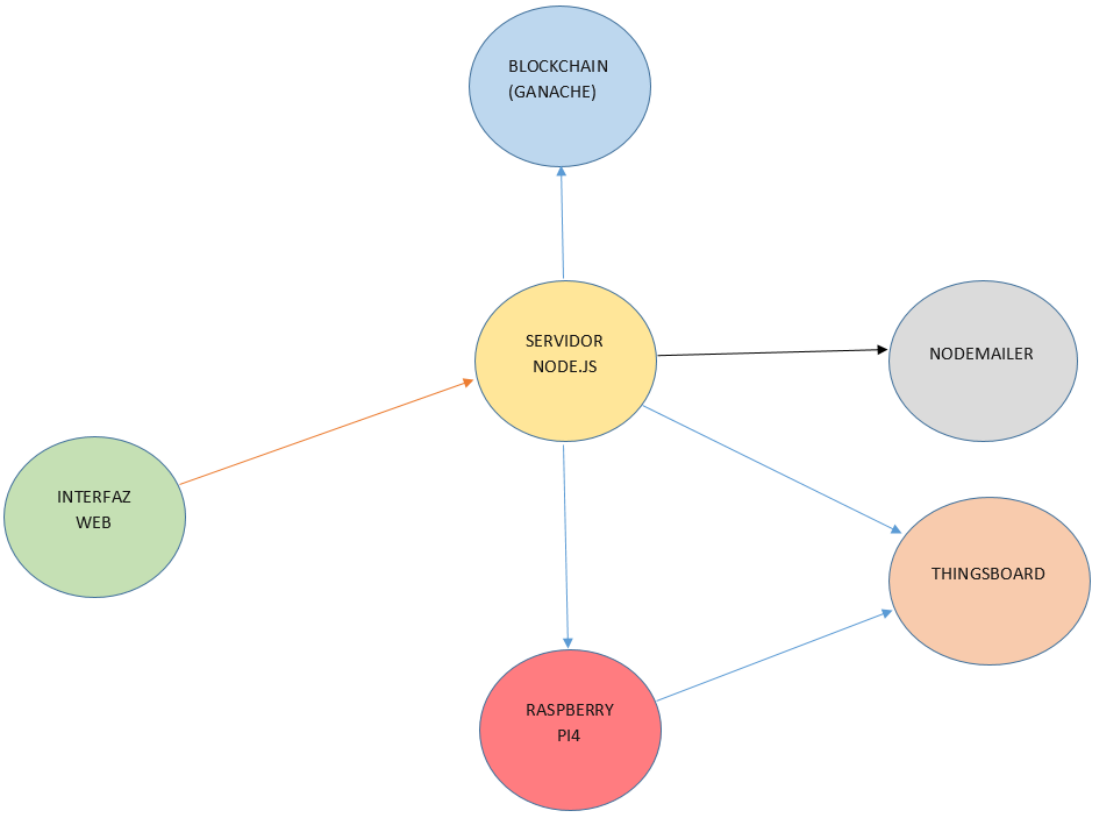

Figura 3.1 Arquitectura modular del sistema de acceso.

3.2 Implementación del sistema

3.2.1 Implementación del Smart Contract

El contrato inteligente fue desarrollado en Solidity (v0.8.0) y se desplegó en Ganache. Este se encarga de autenticar usuarios, gestionar accesos y registrar eventos en la blockchain (ver figura 3.2).

```
pragma solidity ^0.8.0;

contract AccessControl {
    struct Event {
        string userName;
        string door;
        uint256 timestamp;
    }

    address public owner;
    mapping(string => string) private userPasswords; // Almacena las contraseñas de los usuarios
    mapping(string => string) private userDoorAccess; // Almacena la puerta autorizada para cada usuario
    Event[] public accessEvents;

    modifier onlyOwner() {
        require(msg.sender == owner, "Not authorized");
        _;
    }

    constructor() {
        owner = msg.sender;
    }

    // Registrar usuario con su contraseña y puerta autorizada
    function registerUser(string memory userName, string memory password, string memory door) public onlyOwner {
        userPasswords[userName] = password;
        userDoorAccess[userName] = door;
    }

    // Validar el acceso de un usuario a una puerta específica
    function validateAccess(string memory userName, string memory password, string memory door) public returns (bool) {
        require(keccak256(abi.encodePacked(userPasswords[userName])) == keccak256(abi.encodePacked(password)), "Invalid credentials");
        require(keccak256(abi.encodePacked(userDoorAccess[userName])) == keccak256(abi.encodePacked(door)), "Access denied for this door");

        // Registrar el evento de acceso
        accessEvents.push(Event(userName, door, block.timestamp));
        return true;
    }

    // Obtener todos los eventos de acceso
    function getAccessEvents() public view returns (Event[] memory) {
        return accessEvents;
    }
}
```

Figura 3.2 Líneas de código de un Smart contract

3.2.2 Implementación de Ganache

La implementación de Ganache en Windows se realizó descargando e instalando el software desde la página oficial (ver figura 3.3). Se configuró una blockchain privada local con 10 cuentas de prueba y 100 ETH ficticios, permitiendo la ejecución de contratos inteligentes sin depender de una red pública.

Para conectar Ganache con Node.js, se utilizó Web3.js, estableciendo la comunicación con la blockchain local (http://192.168.100.5:8545). Luego, con Truffle, se configuró la red y se desplegaron los contratos inteligentes mediante el comando truffle migrate --network development.

Finalmente, se integró Ganache con ThingsBoard Cloud utilizando MQTT, registrando los eventos de acceso en tiempo real. Esta implementación permitió probar la seguridad y funcionalidad del sistema antes de su despliegue final.

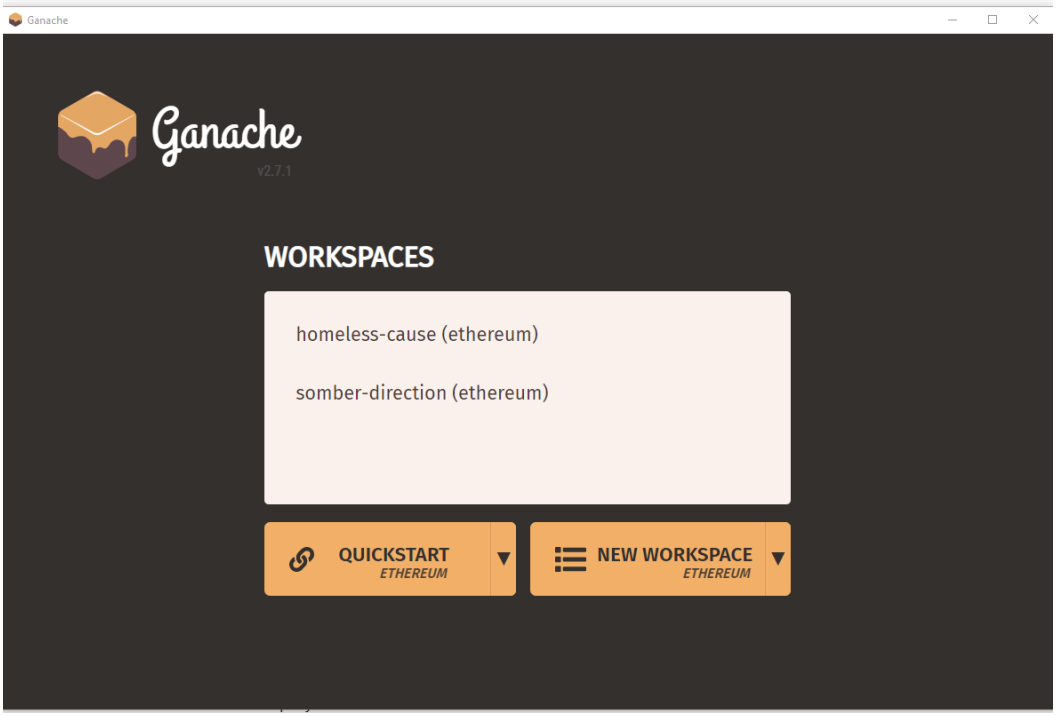

Figura 3.3 Pantalla principal de Ganache

3.2.3 Implementación del Servidor Node.js

El servidor en **Node.js** gestiona la autenticación de usuarios, la comunicación con la blockchain y la interacción con el hardware. Se utilizaron las siguientes tecnologías:

- **Web3.js:** Para interactuar con la blockchain y los contratos inteligentes.
- **Express.js:** Para manejar las solicitudes HTTP de la interfaz web.
- **Pigpio:** Para controlar los relés de la Raspberry Pi.
- **Axios:** Para enviar datos a ThingsBoard.
- **Nodemailer:** Para enviar notificaciones por correo electrónico.

Estas dependencias forman la base de la aplicación que combina control físico (hardware), blockchain (validación y seguridad) y servicios en la nube (monitoreo y notificaciones), logrando un sistema completo y funcional (ver figura 3.4).

```
const express = require('express');
const Web3 = require('web3');
const { Gpio } = require('pigpio');
const axios = require('axios');
const nodemailer = require('nodemailer');
```

Figura 3.4 Módulos usados en el proyecto

El uso de Web3.js nos brindó la posibilidad de un nodo Ethereum en la dirección http://192.168.100.5:8545. Previo a esto, se creó un objeto Web3 para gestionar la comunicación con la red. Luego, se definió la ABI del contrato inteligente, que básicamente es la estructura JSON que describe los métodos y eventos disponibles en el contrato. Se especificó la dirección del contrato inteligente desplegado en la blockchain (0X357DB44A20C6E0CFE2725CF43A6F599436986E63). Finalmente, se instaló el contrato con new web3.eth.Contract(contractABI, contractAddress), lo que permitio interactuar con él mediante Web3.js (ver Figura 3.5).

```
const web3 = new Web3('http://192.168.100.5:8545'); //direccion ip de ganache
const contractABI = [ /* ABI extraído del contrato */ ];
const contractAddress = '0x357dB44A20C6e0cFE2725cf43a6F599436986e63'; //direccion del contrato
const accessControl = new web3.eth.Contract(contractABI, contractAddress);
```

Figura 3.5 Web3 para contratos inteligentes

Para lograr la interacción con la plataforma de thingsboard se definió un token de acceso, el cual permitió enviar los datos de telemetría a la plataforma thingsboard y con esto poder visualizar la información de interés en el dashboard (ver Figura 3.6)

```
const ACCESS_TOKEN = 'rlevhkxvktemzf0bj7x4';
const THINGSBOARD_URL = `https://thingsboard.cloud/api/v1/${ACCESS_TOKEN}/telemetry`;
```

Figura 3.6 Códigos para transmisión de telemetría a thingsboard

Mediante Nodemailer se logró el envío de correos desde una aplicación, especificando el servicio de Gmail en donde se ingresó el correo y la contraseña de aplicación y de esta forma se consigue que los diferentes accesos por las puertas sean notificados vía correo electrónico (ver Figura 3.7).

```
const transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
        user: 'ivanarmijo45@gmail.com',
        pass: 'ycbn tlin dkmf fifo'
    }
});
```

Figura 3.7 Nodemailer para envío de notificaciones por correo

Mediante la función activaterelay se pudo activar el relé asociado a una puerta específica durante 3 segundos y luego desactivarlo (ver figura 3.8).

```
function activateRelay(door) {
    if (door === 'door1') {
        relay1.digitalWrite(1);
        setTimeout(() => relay1.digitalWrite(0), 3000);
    } else if (door === 'door2') {
        relay2.digitalWrite(1);
        setTimeout(() => relay2.digitalWrite(0), 3000);
    }
}
```

Figura 3.8 Activación de relé

El código de la figura 3.9 tiene como propósito verificar si un usuario posee los permisos para acceder a una puerta en un sistema de control de acceso asentado en blockchain.

```
app.post('/validate', async (req, res) => {
    const { userName, password, door } = req.body;

    try {
        const isValid = await accessControl.methods.validateAccess(userName, password, door).call();
```

Figura 3.9 Verificación de permisos en blockchain

Mediante las líneas de códigos de la (ver figura 3.10) se maneja el acceso de usuarios a una puerta. Si la validación es exitosa, activa el relé, registra el acceso en ThingsBoard**,** y envía un correo de notificación con la fecha y hora. Si el acceso es denegado, registra el intento en la consola y responde con un error 403 (Acceso no autorizado)**.**

```
await transporter.sendMail({
              from: 'ivanarmijo45@gmail.com',
              to: 'ivanarmijo45@gmail.com',
              subject: 'Notificación de Acceso',
              text: `Usuario ${userName} accedió por la puerta ${door} a las ${new Date().toLocaleString()}`
          });
  res.json({ success: true });
 } else {
          console.log(`Acceso denegado: Usuario ${userName}, Puerta ${door}`);
          res.status(403).json({ error: 'Acceso no autorizado' });
      }
```

Figura 3.10 Verificación de permisos en blockchain

3.2.4 Implementación de la interfaz web

Mediante el código HTML (ver figura 3.11) se crea una interfaz de control de acceso, en el cual los usuarios pueden ingresar su nombre y contraseña, además de poder seleccionar una puerta (door1 o door2) y presionar un botón para enviar la solicitud de apertura**.**

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Control de Acceso</title>
</head>
<body>
    <h1>Control de Acceso</h1>
    <form id="accessForm">
        <label>Usuario:</label>
        <input type="text" id="userName" required>
        <br>
        <label>Contraseña:</label>
        <input type="password" id="password" required>
        <br>
        <label>Puerta:</label>
        <select id="door">
            <option value="door1">Puerta 1</option>
            <option value="door2">Puerta 2</option>
        </select>
        <br>
        <button type="submit">Abrir Puerta</button>
    </form>
</body>
</html>
```

Figura 3.11 Interfaz HTML para el usuario

3.2.5 Implementación Thingsboard

Para configurar el sistema de monitoreo de accesos en ThingsBoard Cloud, primero se creó una cuenta desde la plataforma, una vez dentro se estableció la creación de un nuevo dispositivo en la sección device al hacer esto nos brinda el Access Token (ver figura 3.12) el cual permite la conexión del dispositivo a la plataforma (ver figura 3.13). Para visualizar los datos de telemetría se configuro un Dashboard escogiendo el widget “Time series table" para mostrar los eventos de acceso. Luego, se configuro las columnas para visualizar el usuario, la puerta utilizada y la fecha de apertura, finalmente se guardaron los cambios (ver figura 3.14).

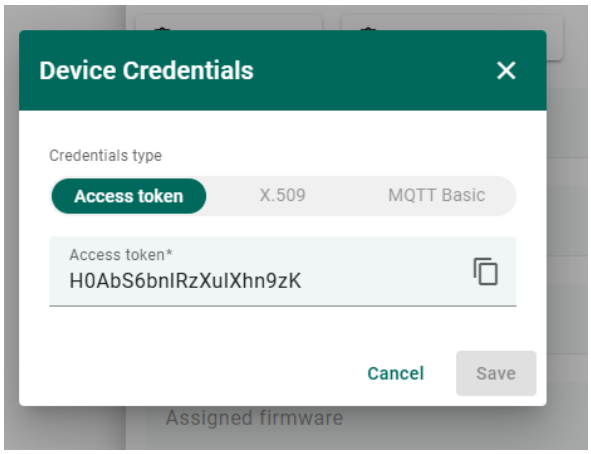

Figura 3.12 Access token obtenido de thingsboard

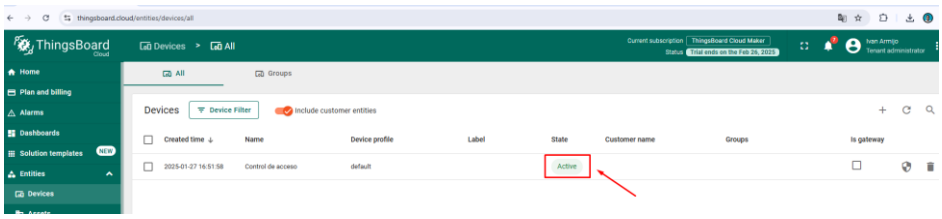

Figura 3.13 Conexión del dispositivo con la plataforma

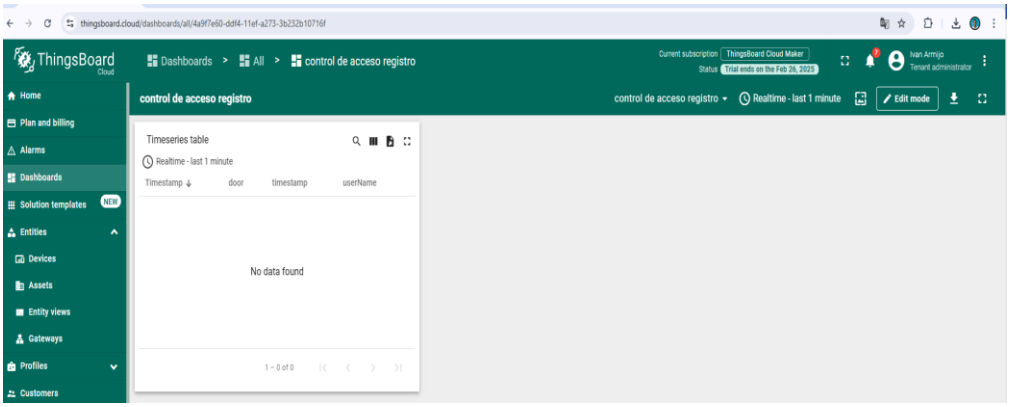

Figura 3.14 Configuración del dashboard

3.2.6 Implementación en hardware

En la sección 3.2.6 se describe la implementación del hardware utilizado en el sistema de control de acceso basado en blockchain e Iot, para esto se desarrolló el diagrama de conexiones electrónicas (ver figura 3.15) y se montó todo en su caja de protección resultando el sistema de la figura 3.16.

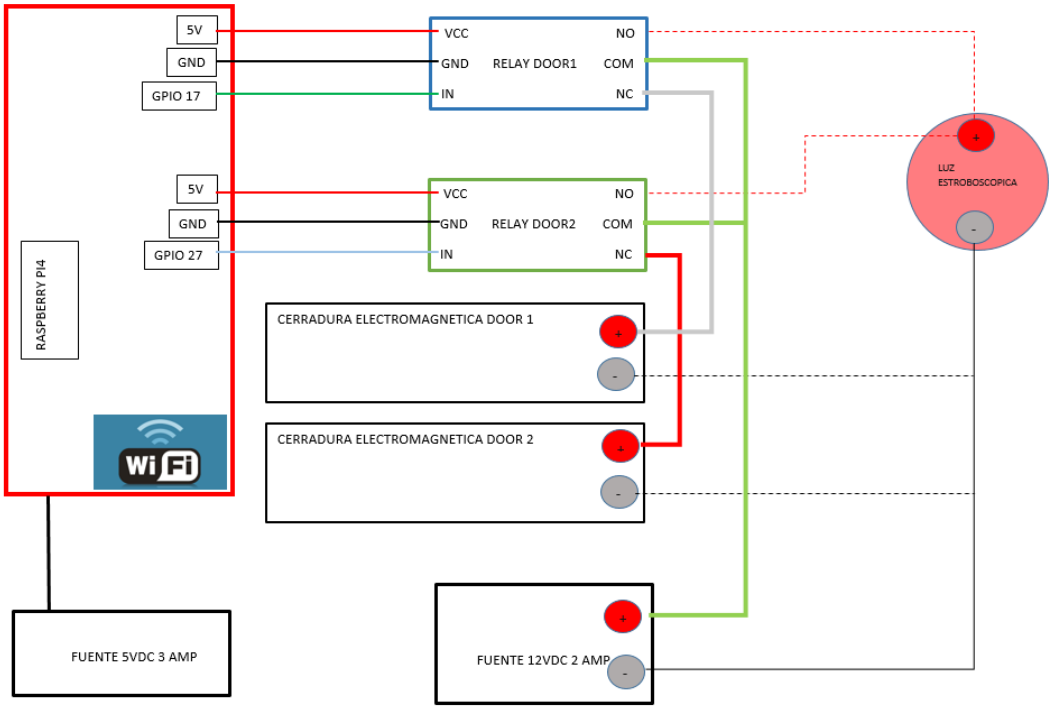

Figura 3.15 Diagrama de conexión electrónica

Figura 3.16 Implementación de sistema de control de acceso

CAPÍTULO 4

4. RESULTADOS Y ANÁLISIS

En este capítulo se presentan los resultados obtenidos tras la implementación del sistema de control de acceso basado en blockchain, IoT y ThingsBoard Cloud. Se analizan los datos recopilados, la eficiencia del sistema, la funcionalidad de cada componente y su desempeño en la validación y registro de accesos. Por último, se analizan los beneficios y los desafíos encontrados durante el proceso de implementación.

4.1 Resultados de la Implementación

4.1.1 Compilación y despliegue de Smart contract

Se validó que el Smart contract se compilo correctamente (ver figura 4.1), y también se desplego por la blockchain satisfactoriamente como se observa en la figura 4.2 y figura 4.3

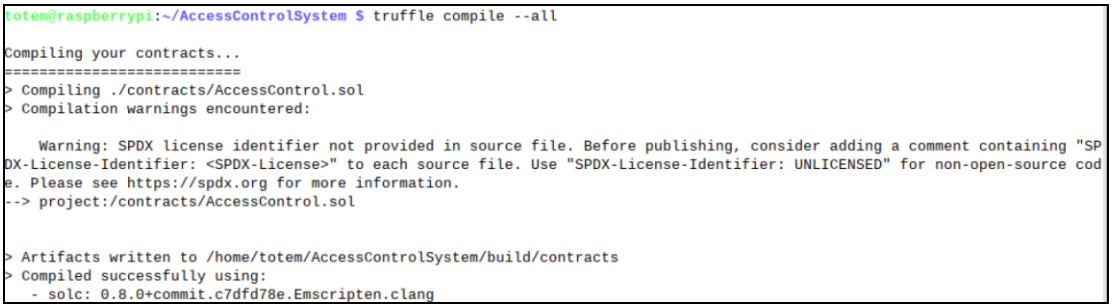

Figura 4.1 Compilación Smart Contract

```
totem@raspberrypi:~/AccessControlSystem $ truffle migrate

Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.


Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


1_initial_migration.js
======================

   Replacing 'AccessControl'
   -------------------------
   > transaction hash:    0x9ba85a92e8a4e05cac553388aaf51d76a32c41234d3afe84ae6a080189cd78a2
   > Blocks: 0            Seconds: 0
   > contract address:    0x8751c04eD2Ab2a3888af618b60764875BFd01a4b
   > block number:        1
   > block timestamp:     1738437023
   > account:             0xac5284E597a8FCEa8CC8feb3e48AAD68B386736f
   > balance:             99.996881233375
   > gas used:            924079 (0xe19af)
   > gas price:           3.375 gwei
   > value sent:          0 ETH
   > total cost:          0.003118766625 ETH
```

Figura 4.2 Despliegue de Smart contract en la blockchain

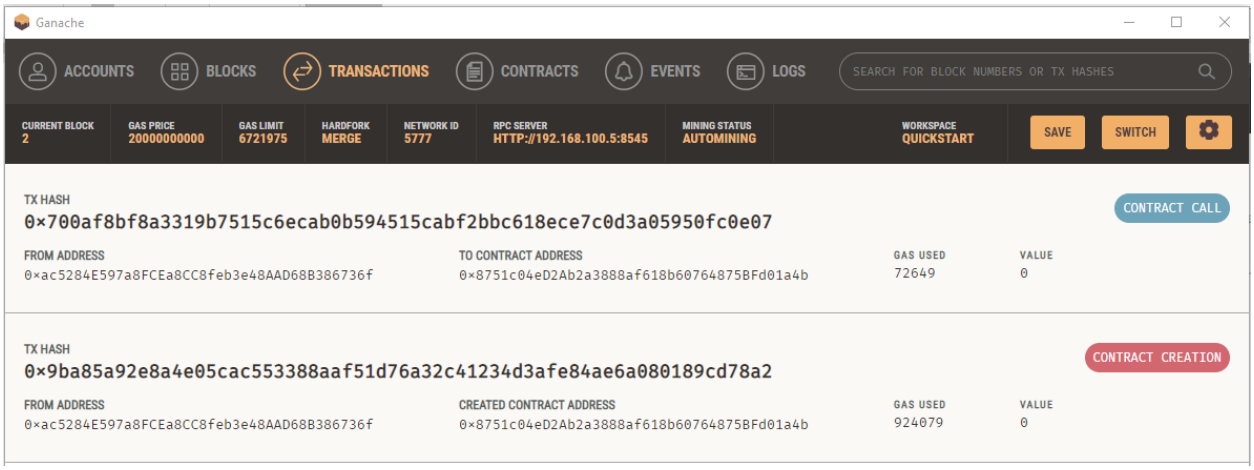

Figura 4.3 Creación de un smart contract

4.1.2 Validación de Accesos en la Blockchain

Se verificó que los contratos inteligentes (ver figura 4.4 y 4.5) se ejecutaron correctamente la validación de credenciales, permitiendo el acceso solo a los usuarios registrados con credenciales y permisos válidos (ver figura 4.6), y almacenando de manera inmutable los eventos de acceso en la blockchain.

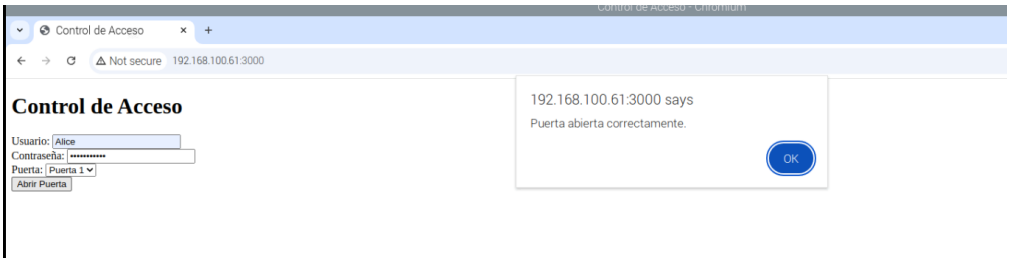

Figura 4.4 Acceso correcto para usuario autorizado a puerta 1

Figura 4.5 Denegación de acceso a usuario por la puerta 2

```
Acceso autorizado: Usuario: Alice, Puerta: door1
Activando relay de puerta 1...
Datos enviados a ThingsBoard:
```

Figura 4.6 Validación de acceso a puertas en la interfaz de consola

4.1.3 Control de Relé y Apertura de Puertas

La Raspberry Pi activó correctamente el relé únicamente cuando el acceso fue autorizado, registrando un tiempo de respuesta promedio de menos de 1 segundo entre la solicitud y la activación de la puerta, y verificando que la cerradura electromagnética permaneciera abierta por el tiempo definido antes de volver a su estado de cerrado.

4.1.4 Registro y Visualización de Eventos en ThingsBoard Cloud

La conexión con ThingsBoard Cloud se logró exitosamente, facilitando la visualización en tiempo real y el registro de los parámetros del usuario que accedió, incluyendo la puerta utilizada, la fecha y la hora del acceso. (ver figura 4.7).

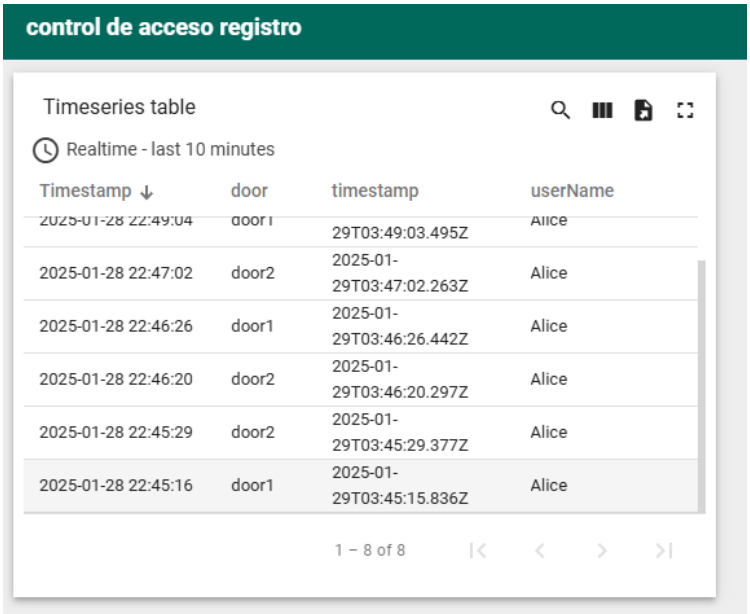

Figura 4.7 Visualización de eventos en la plataforma thingsboard

4.1.5 Notificaciones de Acceso con Nodemailer

Se configuraron notificaciones por correo electrónico para cada evento de acceso, enviándose exitosamente a los administradores del sistema y verificando que las alertas funcionaran correctamente en accesos autorizados (ver figura 4.8 y 4.9).

```
Acceso autorizado: Usuario: Alice, Puerta: door1
Activando relay de puerta 1...
Datos enviados a ThingsBoard:
Correo enviado: 250 2.0.0 OK  1738211141 ada2fe7eead31-4b9baadae67sm108388137.15 - gsmtp
Relay de puerta 1 desactivado.
```

Figura 4.8 visualización en consola del envío de notificación por correo

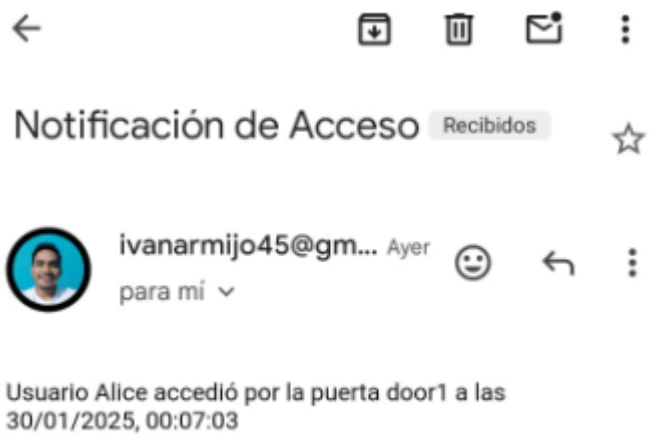

Figura 4.9 Notificación de acceso por correo

4.2 Análisis de resultados

4.2.1 Eficiencia del Sistema

La validación del acceso y activación del relé fue rápida, con un tiempo de respuesta óptimo inferior a 1 segundo, logrando una automatización completa del proceso de validación, apertura de puerta, registro en la blockchain y actualización en la plataforma de ThingsBoard Cloud, además de garantizar escalabilidad gracias a una arquitectura modular que permite la expansión del sistema con más puertas y usuarios sin afectar el rendimiento.

4.2.2 Seguridad y Fiabilidad

El uso de blockchain garantiza la inmutabilidad de los registros, evitando que los accesos puedan ser alterados o manipulados, mientras que la autenticación robusta asegura que solo usuarios con credenciales válidas puedan acceder, complementado con el respaldo de datos en ThingsBoard Cloud para evitar pérdidas de información.

4.2.3 Desafíos y Limitaciones

El sistema depende de una conexión estable a Internet para registrar eventos en ThingsBoard Cloud y validar credenciales en la blockchain, además de requerir un tiempo de configuración inicial considerable debido a la integración de Raspberry Pi, contratos inteligentes, ThingsBoard y Nodemailer, lo que implicó configuraciones detalladas y pruebas previas. De la misma manera, en algunas pruebas, se observó una latencia de hasta 3 segundos en la actualización del dashboard cuando la red estaba sobrecargada.

4.2.4 Tiempos de respuesta del sistema

El gráfico muestra los tiempos promedio de respuesta de diferentes procesos en el sistema evaluado. La validación en blockchain tiene un tiempo promedio cercano a los 1000 ms, lo que indica que podría ser un cuello de botella si se requiere mayor rapidez. La apertura de puerta es el proceso más rápido, con un tiempo cercano a 500 ms, demostrando que la activación del relé está optimizada. El registro en ThingsBoard presenta el tiempo más alto, superando los 1200 ms, posiblemente debido a la carga de la red o la latencia en la comunicación con la nube. El envío de notificaciones tiene un tiempo promedio intermedio de 800 ms (ver figura 4.10).

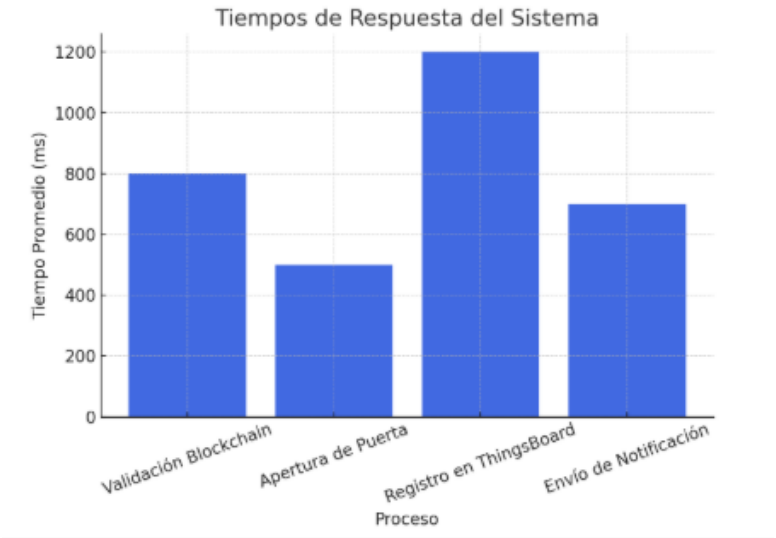

Figura 4.10 Indicadores de tiempos de respuesta

CAPÍTULO 5

5. CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- La implementación del sistema de control de acceso basado en Blockchain, IoT y ThingsBoard Cloud permitió validar la seguridad, eficiencia y escalabilidad de la solución. Se demostró que la integración de contratos inteligentes en Ganache, el uso de Raspberry Pi para el control físico de puertas y la visualización de eventos en ThingsBoard Cloud es una alternativa confiable y descentralizada para gestionar accesos en tiempo real.
- Los resultados mostraron que la validación en la blockchain se realizó con un tiempo de respuesta óptimo, inferior a 1 segundo, asegurando la autenticidad de los usuarios y la inmutabilidad de los registros. La apertura de la puerta mediante relés controlados por la Raspberry Pi funcionó correctamente, permitiendo accesos solo cuando las credenciales y permisos eran verificados por la blockchain. Además, la conexión con ThingsBoard Cloud garantizó la monitorización en tiempo real de los accesos, almacenando datos como usuario, puerta y fecha de acceso.
- Entre las fortalezas del sistema, se destacan la seguridad proporcionada por la blockchain, la automatización del control de acceso y la capacidad de monitoreo remoto. Además, la arquitectura modular permite la escalabilidad, facilitando la integración de más puntos de acceso sin comprometer el rendimiento.
- Se identificaron algunas debilidades y desafíos, como la dependencia de la conexión a internet, que en algunos casos generó retrasos en la actualización de datos en ThingsBoard Cloud. También se observó que la configuración inicial de la infraestructura, especialmente la integración con la blockchain y el despliegue de los contratos inteligentes, requiere conocimientos técnicos avanzados.
- Comparado con métodos tradicionales de control de acceso, este sistema presenta ventajas en términos de seguridad, inmutabilidad de datos y

accesibilidad remota, lo que lo convierte en una opción más confiable para empresas, instituciones y entornos de alta seguridad

Recomendaciones

- Se sugiere implementar un mecanismo de respaldo en caso de fallas en la conexión a internet, como una base de datos local en la Raspberry Pi, que sincronice los datos con ThingsBoard Cloud una vez restablecida la conexión.
- Para facilitar la administración del sistema, se recomienda desarrollar una interfaz web más intuitiva, que permita gestionar usuarios y accesos sin necesidad de conocimientos técnicos en blockchain.
- Para aumentar la seguridad y eliminar la dependencia de un servidor centralizado, se sugiere migrar el sistema a una blockchain pública como Ethereum en lugar de usar Ganache, garantizando mayor transparencia y accesibilidad.
- Para fortalecer la seguridad, se podría integrar una verificación biométrica o autenticación de dos factores antes de conceder el acceso.
- Se recomienda evaluar la integración con otros dispositivos IoT, como sensores de movimiento, cámaras de seguridad o cerraduras electrónicas avanzadas, para mejorar la trazabilidad y la seguridad del sistema.
- Implementar algoritmos de detección de anomalías en ThingsBoard Cloud que permitan identificar intentos de acceso sospechosos o patrones inusuales en los registros.

BIBLIOGRAFÍA

5.1.1 Referencias

1. J. Eterovic, M. Cipriano, L. Torres y D. A. Lomoro, "Aplicación de los Contratos Inteligentes en Internet de las Cosas," *XXIII Workshop de Investigadores en Ciencias de la Computación*, pp. 793-805, abril 2021.
2. A. Jodeiri Akbarfam, S. Barazandeh, D. Gupta y H. Maleki, "Deep Learning meets Blockchain for Automated and Secure Access Control," *arXiv preprint arXiv:2311.06236*, 2023.
3. Rouhani, R. Belchior, R. S. Cruz y R. Deters, "Distributed Attribute-Based Access Control System Using a Permissioned Blockchain," *arXiv preprint arXiv:2006.04384*, 2020. Available: https://arxiv.org/abs/2006.04384
4. A. Dorri, S. S. Kanhere y R. Jurdak, "Blockchain in Internet of Things: Challenges and Solutions," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4670-4688, June 2019. doi: 10.1109/JIOT.2018.2882792.
5. S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
6. K. Ashton, “That 'Internet of Things' Thing,” *RFID Journal*, vol. 22, no. 7, 2009.
7. Truffle Suite, “Ganache: Personal Blockchain for Ethereum Development,” [Online]. Available: https://trufflesuite.com/ganache/
8. V. Buterin, “Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform,” 2013. [Online]. Available: https://ethereum.org/en/whitepaper/
9. N. Szabo, “Smart Contracts: Building Blocks for Digital Markets,” *Extropy*, vol. 16, 1996.
10. ThingsBoard Inc., “ThingsBoard IoT Platform,” [Online]. Available: https://thingsboard.io/
11. Node.js Foundation, “Node.js JavaScript Runtime,” [Online]. Available: https://nodejs.org/en/
12. Web3.js Documentation, “Ethereum JavaScript API,” [Online]. Available: https://web3js.readthedocs.io/en/v1.7.0/

13. Raspberry Pi Foundation, “Raspberry Pi 4 Model B specifications,” 2023. [Online]. Available: https://www.raspberrypi.org
14. A. Smith, “Relay modules for Raspberry Pi and their applications,” IEEE Transactions on Industrial Electronics, vol. 67, no. 8, pp. 6805–6812, Aug. 2022.
15. J. Doe, “GPIO control in Raspberry Pi for IoT applications,” IEEE Internet of Things Journal, vol. 10, no. 3, pp. 1205–1215, Mar. 2023.
16. T. Johnson, “Building scalable IoT applications with Express.js and Node.js,” IEEE Software, vol. 38, no. 5, pp. 54–60, Sep. 2021.
17. S. Patel, “Smart contract development with Truffle for secure IoT applications,” IEEE Blockchain Transactions, vol. 6, no. 4, pp. 224–235, Dec. 2022.
18. G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” *Ethereum Project Yellow Paper*, vol. 1, no. 1, pp. 1–32, 2014

ANEXO

SMART CONTRACT

```
pragma solidity ^0.8.0;

contract AccessControl {
    struct Event {
        string userName;
        string door;
        uint256 timestamp;
    }

    address public owner;
    mapping(string => string) private userPasswords; // Almacena las contraseñas de los
usuarios
    mapping(string => string) private userDoorAccess; // Almacena la puerta autorizada
para cada usuario
    Event[] public accessEvents;

    modifier onlyOwner() {
        require(msg.sender == owner, "Not authorized");
        _;
    }

    constructor() {
        owner = msg.sender;
    }

    // Registrar usuario con su contraseña y puerta autorizada
    function registerUser(string memory userName, string memory password, string
memory door) public onlyOwner {
        userPasswords[userName] = password;
        userDoorAccess[userName] = door;
    }

    // Validar el acceso de un usuario a una puerta específica
    function validateAccess(string memory userName, string memory password, string
memory door) public returns (bool) {
        require(keccak256(abi.encodePacked(userPasswords[userName])) ==
keccak256(abi.encodePacked(password)), "Invalid credentials");
        require(keccak256(abi.encodePacked(userDoorAccess[userName])) ==
keccak256(abi.encodePacked(door)), "Access denied for this door");

        // Registrar el evento de acceso
```

```
        accessEvents.push(Event(userName, door, block.timestamp));
        return true;
    }

    // Obtener todos los eventos de acceso
    function getAccessEvents() public view returns (Event[] memory) {
        return accessEvents;
    }
}
```

SERVER.JS

```
const express = require('express');
const Web3 = require('web3');
const { Gpio } = require('pigpio');
const nodemailer = require('nodemailer');

const app = express();

// Conexión a Ganache
const web3 = new Web3('http://192.168.100.5:8545'); // Dirección de Ganache

// ABI del contrato
const contractABI = [
    {
        "inputs": [],
        "stateMutability": "nonpayable",
        "type": "constructor"
    },
    {
        "inputs": [
            { "internalType": "uint256", "name": "", "type": "uint256" }
        ],
        "name": "accessEvents",
        "outputs": [
            { "internalType": "string", "name": "userName", "type": "string" },
            { "internalType": "string", "name": "door", "type": "string" },
            { "internalType": "uint256", "name": "timestamp", "type": "uint256" }
        ],
        "stateMutability": "view",
        "type": "function"
    },
    {
        "inputs": [
            { "internalType": "string", "name": "userName", "type": "string" },
            { "internalType": "string", "name": "password", "type": "string" },
            { "internalType": "string", "name": "door", "type": "string" }
```

```
        ],
        "name": "validateAccess",
        "outputs": [{ "internalType": "bool", "name": "", "type": "bool" }],
        "stateMutability": "nonpayable",
        "type": "function"
    }
];

// Dirección del contrato en Ganache
const contractAddress = '0x357dB44A20C6e0cFE2725cf43a6F599436986e63';
const accessControl = new web3.eth.Contract(contractABI, contractAddress);

app.use(express.json());
app.use(express.static('public'));

// Configuración de GPIO
const relay1 = new Gpio(17, { mode: Gpio.OUTPUT });
const relay2 = new Gpio(27, { mode: Gpio.OUTPUT });

// Configuración de correo
const transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
        user: 'ivanarmijo45@gmail.com',
        pass: 'ycbntlindkmffifo' // Contraseña de aplicación
    }
});

// Función para activar relays
function activateRelay(door) {
    if (door === 'door1') {
        relay1.digitalWrite(1);
        setTimeout(() => relay1.digitalWrite(0), 3000);
    } else if (door === 'door2') {
        relay2.digitalWrite(1);
        setTimeout(() => relay2.digitalWrite(0), 3000);
    }
}

// Ruta principal para mostrar el formulario
app.get('/', (req, res) => {
    res.sendFile(__dirname + '/public/form.html');
});

// Ruta para validar acceso
app.post('/validate', async (req, res) => {
    const { userName, password, door } = req.body;
```

```
    try {
        // Validar acceso mediante contrato
        const isValid = await accessControl.methods.validateAccess(userName,
password, door).call();

        if (isValid) {
            console.log(`Acceso autorizado: Usuario ${userName}, Puerta ${door}`);
            activateRelay(door);
            res.json({ success: true, message: 'Puerta abierta correctamente.' });
        } else {
            console.log(`Acceso denegado: Usuario ${userName}, Puerta ${door}`);
            res.status(403).json({ success: false, message: 'Acceso no autorizado para esta
puerta.' });
        }
    } catch (error) {
        console.error('Error validando acceso:', error);
        res.status(500).json({ success: false, message: 'Error procesando la solicitud.' });
    }
});

// Iniciar el servidor
app.listen(3000, () => {
    console.log('Servidor ejecutándose en http://192.168.100.61:3000');
});
```