ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

CREACIÓN DE UN PROTOTIPO DE SISTEMA DE IDENTIFICACIÓN ESTUDIANTIL UTILIZANDO TECNOLOGÍA BLOCKCHAIN.

COMPONENTE BACKEND DEL PROTOTIPO DE SISTEMA DE IDENTIFICACIÓN ESTUDIANTIL BASADO EN BLOCKCHAIN.

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO/A EN CIENCIAS DE LA COMPUTACIÓN

SERGIO SALVADOR NARANJO SANCHEZ

sergio.naranjo@epn.edu.ec

DIRECTOR: SANG GUUN YOO, Ph.D.

sang.yoo@epn.edu.ec

DMQ, febrero de 2024

CERTIFICACIONES

Yo, SERGIO SALVADOR NARANJO SANCHEZ declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

_____

SERGIO SALVADOR NARANJO SANCHEZ

Certifico que el presente trabajo de integración curricular fue desarrollado por SERGIO SALVADOR NARANJO SANCHEZ, bajo mi supervisión.

_____

SANG GUUN YOO, Ph.D

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

SERGIO SALVADOR NARANJO SANCHEZ

SANG GUUN YOO

CARLOS ALEJANDRO ESTRADA RIVERA

VERONICA JACQUELINE TOASA ROCHA

DEDICATORIA

A mi familia que me ha enseñado a continuar a pesar de las dificultades.

AGRADECIMIENTO

A mis padres por estar siempre presentes y apoyarme a seguir adelante. Por todo el amor y sacrificio que han hecho por mí y por mis hermanos. Admiro su fuerza, su determinación y su capacidad de sobreponerse a las dificultades.

A mis hermanos por siempre apoyarme en los buenos y malos momentos, por levantarme cuando caigo, por animarme y mostrarme que puedo superar cualquier obstáculo si me lo propongo.

Al Profe Sang, por darme la oportunidad y confiar en mi para llevar a cabo este proyecto.

ÍNDICE DE CONTENIDO

RESUMEN

Blockchain es una de las nuevas tecnologías más revolucionarias por su enfoque descentralizado y seguro para el registro y verificación de transacciones, su principal aplicación son las criptomonedas, pero en los últimos años se ha expandido el uso de esta tecnología a diferentes campos de aplicación. Blockchain es una cadena de bloques de información descentralizada, es decir que la información está distribuida entre múltiples participantes dentro de una red, donde cada uno de los participantes tiene una copia de la cadena de bloques. Los miembros de la red se ponen de acuerdo sobre el estado de la cadena de bloques mediante un protocolo de consenso, eliminando de esta forma la necesidad de una autoridad central.

Este documento presenta la información del desarrollo de un prototipo de sistema de identificación estudiantil basado en la tecnología blockchain, que aborda la gestión de los datos de identificación, la seguridad y la integridad de la información. El componente desarrollado es este trabajo consta de la parte backend del prototipo, es decir se crea una red blockchain Hyperledger Besu con el protocolo de consenso IBFT 2.0, además se crea la logia del prototipo mediante los Smart Contracts creados con Solidity.

El desarrollo del prototipo muestra como la blockchain puede ser integrado de manera efectiva en el ambiente educativo, proporcionando soluciones eficientes en los sistemas de identificación estudiantil, ofreciendo métodos más seguros y transparentes.

PALABRAS CLAVE: blockchain, identificación, protocolos de consenso.

ABSTRACT

Blockchain is one of the most revolutionary new technologies for its decentralized and secure approach to record and verify transactions, its main application is cryptocurrencies, but in recent years the use of this technology has expanded to different fields of application. Blockchain is a blockchain of decentralized information, meaning that the information is distributed among multiple participants within a network, where each participant has a copy of the blockchain. Network members agree on the state of the blockchain through a consensus protocol, thus eliminating the need for a central authority.

This paper presents information from the development of a prototype student identification system based on blockchain technology, which addresses identification data management, security and information integrity. The component developed is this work consists of the backend part of the prototype, i.e. a Hyperledger Besu blockchain network is created with the IBFT 2.0 consensus protocol, in addition the prototype's lodge is created using Smart Contracts created with Solidity.

The development of the prototype shows how blockchain can be effectively integrated into the educational environment, providing efficient solutions in student identification systems, offering more secure and transparent methods.

KEYWORDS: blockchain, identification, consensus protocols.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El componente desarrollado es un prototipo que permite la identificación estudiantil basada en la tecnología blockchain. El prototipo se compone de varias partes fundamentales, las cuales desempeñan un papel crucial en el funcionamiento del sistema. La arquitectura del prototipo se muestra en la Figura 1.

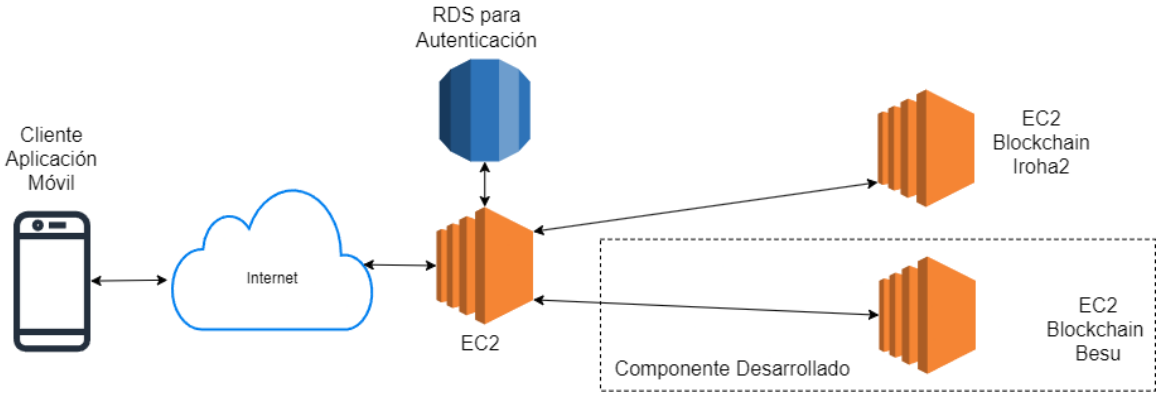


Figura 1 Arquitectura General del prototipo

El prototipo de sistema de identificación estudiantil basado en Blockchain consta de tres componentes. Los dos primeros son dos implementaciones del backend del prototipo donde se crea la red blockchain y se le da la lógica al prototipo; estas implementaciones son desarrolladas dentro del servicio de EC2 de AWS, en la cual la primera es implementada mediante Iroha2, y el segundo mediante el Hyperledger Besu. El último componente consiste en la aplicación móvil y su lógica para interactuar con las implementaciones de blockchain y que permita a los usuarios acceder al servicio.

El componente desarrollado en este trabajo es uno de los backend del prototipo de identificación estudiantil que se encarga de gestionar la infraestructura de la red blockchain (ver Figura 1). Esto incluye la administración de la red, la implementación del algoritmo de consenso IBFT 2.0 proof of authority (PoA) y el despliegue de Smart Contracts en Solidity. Los contratos inteligentes son los que codifican la lógica del prototipo y permiten la interacción con la aplicación. Entre las funciones principales se encuentran la creación, verificación y anulación de credenciales de estudiantes.

En primera instancia, se optó por trabajar con Hyperledger Besu como la plataforma blockchain para el prototipo porque se trata de un cliente Ethereum flexible, idóneo tanto para redes privadas como para redes públicas [1]. Esta versatilidad permite adaptar la configuración de la red según las necesidades específicas de cada entidad. Además, la plataforma se destaca por su alto rendimiento en el procesamiento eficiente de transacciones, ideal para aplicaciones que demandan rapidez y baja latencia. Su

escalabilidad robusta garantiza que la red pueda expandirse horizontalmente para satisfacer las crecientes demandas sin sacrificar la confiabilidad [2].

A su vez, Hyperledger Besu se distingue por su capacidad para aprovechar el sólido ecosistema Ethereum, al ser un cliente de esta plataforma. Esto permite a las empresas beneficiarse de la experiencia, seguridad y madurez de Ethereum [1], respaldados por una amplia comunidad de desarrolladores y recursos. La plataforma facilita la creación eficiente de redes privadas, otorgando a las empresas un mayor control sobre su infraestructura y datos, estableciendo reglas específicas y políticas personalizadas [1].

El algoritmo de consenso usado es IBFT 2.0 proof of authority (PoA) consensus protocol, ya que este se destaca por su seguridad y eficiencia. Su tolerancia a fallos permite resistir hasta un tercio de los nodos maliciosos en redes privadas [3]. Cuenta con validadores conocidos, alta velocidad de transacción y compatibilidad con Ethereum, IBFT 2.0 ofrece un protocolo robusto que garantiza la irreversibilidad en transacciones y resuelve conflictos para mantener la consistencia del estado de la red [3].

Este protocolo presenta una elección estratégica al implementar PoA, fortaleciendo la seguridad de IBFT 2.0 en redes privadas blockchain. Con una tolerancia a fallos del 33%, previene ataques Sybil y 51%, haciendo más difícil el control malicioso. La identidad preseleccionada de los validadores y su verificación a través de certificados digitales o KYC refuerzan la confianza en la red, mientras que la eficiencia y escalabilidad se destacan con transacciones rápidas y capacidad de adaptación al aumento de nodos [2].

Se desarrollan los smart contracts en Solidity, estos conforman la parte lógica del prototipo blockchain, un smart contract es un programa que se encuentra dentro de la blockchain que define y automatiza la lógica del prototipo [4]. Para el prototipo desarrollado los smart contracts son los encargados de la interacción con los usuarios, y gestionar la información de identificación estudiantil.

La Figura 2, muestra la arquitectura del componente desarrollado. El componente esta desarrollado dentro de una instancia EC2 de AWS donde se tiene una red Hyperledger Besu con cuatro nodos interconectados entre sí, la red utiliza el protocolo de consenso IBFT 2.0 para llegar a los consensos y crear los bloques, dentro de esta red esta la lógica del prototipo con los contratos inteligentes. La red expone un webservice en el puerto 8545 para que la aplicación del usuario pueda conectarse a la red e interactuar. Además, se tiene también una aplicación Chainlens en el puerto 8081, esta aplicación permite explorar dentro de la red las diferentes transacciones realizadas.

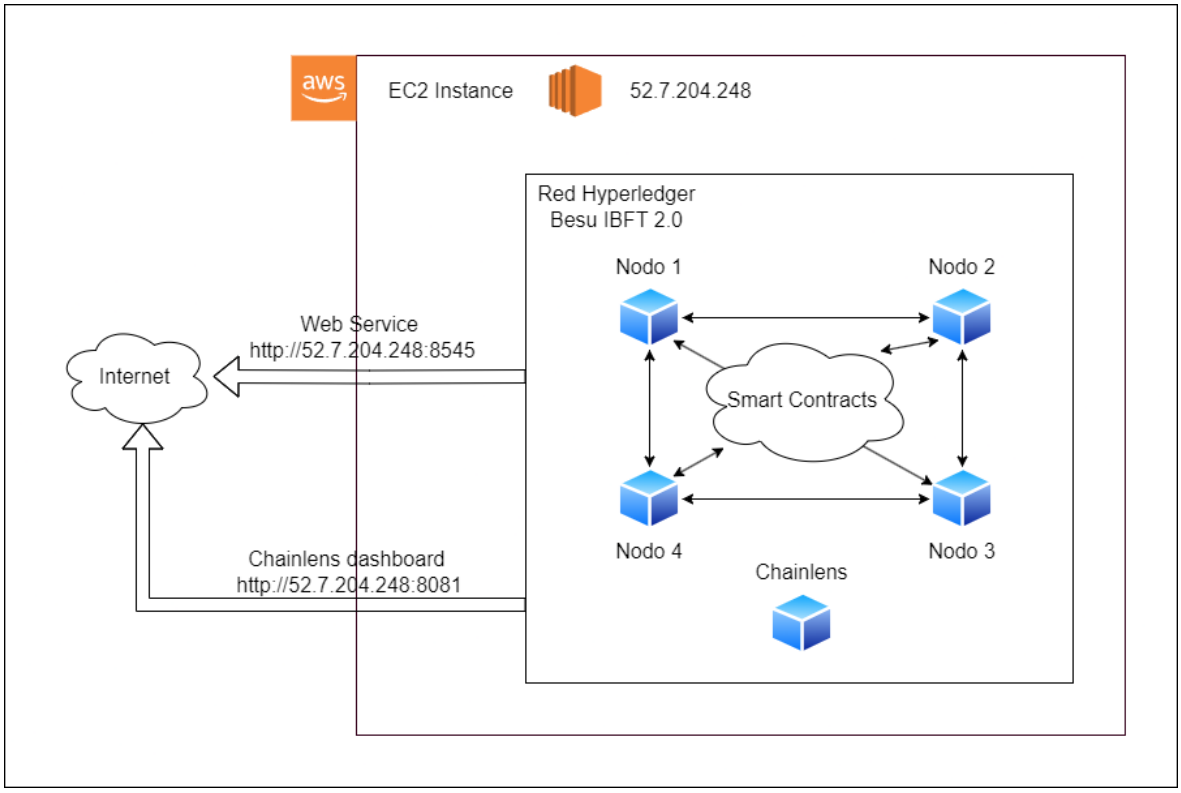


Figura 2 Arquitectura del componente desarrollado.

1.1 Objetivo general

Construir el backend del prototipo de sistema de identificación estudiantil basado en blockchain que comprende la red blockchain y el funcionamiento lógico del mismo.

1.2 Objetivos específicos

1. Entender los requerimientos del sistema de identificación estudiantil basado en la tecnología de blockchain.
2. Crear la red blockchain basado en los requerimientos identificados.
3. Generar la lógica de la aplicación mediante Smart Contracts.
4. Comprobar el funcionamiento del prototipo mediante pruebas de ejecución de los Smart Contracts.

1.3 Alcance

En este componente, se aborda el desarrollo del backend del prototipo de identificación estudiantil basado en blockchain. Este componente no solo se encarga de instaurar la red

blockchain, sino que también se ocupa de establecer la lógica del prototipo mediante la implantación de Smart Contracts. En el aspecto de la creación de la red se limita el número de nodos a cuatro, garantizando con esto una estructura eficiente y manejable. Además, para la creación de la lógica del prototipo se usa la tecnología de Non Fungible Token (NFT) para representar la información de cada estudiante. Como parte integral del sistema, se implementa un servicio web que facilita la conexión y la interacción de los usuarios con la red.

1.4 Marco teórico

Dentro de la siguiente sección se exponen los diferentes conceptos para poder comprender el desarrollo del componente del prototipo de sistema de identificación estudiantil basado en blockchain, así como las diferentes herramientas utilizadas para el mismo.

Blockchain

En los últimos años se ha incrementado la transformación digital del mundo, en este contexto surge la tecnología blockchain, una tecnología descentralizada basada en la criptografía y en la descentralización de datos. En sus inicios, esta tecnología se usó en criptomonedas, como Bitcoin. Sin embargo, por sus características también ha captado la atención de investigadores, empresas y desarrolladores [5].
Blockchain es una tecnología descentralizada que funciona como un registro descentralizado e inmutable de información, la información esta almacenada como transacciones dentro de bloques de información interconectados. Blockchain está basado en algoritmos criptográficos que garantizan la inmutabilidad de la información almacenada. Cada nodo de una red blockchain tiene una copia de la cadena de bloques que se modifican mediante algoritmos de consenso, esto garantiza la transparencia y la integridad de los datos [6].
Existe un crecimiento sostenido en cuanto a documentos de investigación en el campo de blockchain desde el año 2019, algunos de los campos en los que más se usan son los campos de ciberseguridad en aplicaciones relacionadas con el manejo de identidades de usuarios y protección de datos, También existen aplicaciones en otros campos como en IoT, medicina, videojuegos, etc. [7].

Hyperledger Besu

Hyperledger Besu es un cliente Ethereum de código abierto que puede ejecutarse en la red pública de Ethereum o en redes privadas autorizadas, este incluye varios algoritmos de consenso como PoS (Proof of Stake), PoW (Proof of work), PoA (Proof of Authority) y IBFT (Istanbul Byzantine Fault Tolerant) [8].

Dentro de las redes privadas también se tiene la característica de Ethereum de la EVM (Ethereum Virtual Machine), que permite el despliegue y ejecución de contratos inteligentes a través de transacciones dentro de la blockchain [8].

Hyperledger Besu también proporciona APIs accesibles para los usuarios a través de protocolos HTTP y WebSockets, en formato JSON-RPC, tanto para la red pública como para redes privadas, además cuenta con una API GraphQL para visualizar la información de la red [8].

IBFT 2.0 Proof of Autority

Es un protocolo de consenso de blockchain Proof-of-Authority (PoA) Byzantine-Fault-Tolerant (BFT), que permite a las redes privadas aprovechar las capacidades de los contratos inteligentes de Ethereum, garantizan la finalidad inmediata, es robusto y cuenta con un conjunto de validadores dinámicos [9].

El protocolo IBFT 2.0 es un protocolo de consenso blockchain diseñado para abordar las limitaciones de los sistemas centralizados tradicionales. Aprovecha la naturaleza descentralizada de la tecnología de registros distribuidos, donde los datos de transacciones se almacenan en múltiples nodos, evitando alteraciones unilaterales por parte de una autoridad central. El protocolo se centra en métricas clave de rendimiento influenciadas por los protocolos de consenso, como el rendimiento, la latencia y la robustez contra diversos ataques. IBFT 2.0 emplea un enfoque tolerante a fallas bizantinas (BFT) para garantizar la consistencia incluso en presencia de nodos maliciosos, siendo los validadores fundamentales para crear nuevos bloques y mantener la integridad de la cadena de bloques. Además, el mecanismo de Prueba de Autoridad (PoA) del protocolo evita ataques Sybil al conferir derechos de creación de bloques solo a un conjunto definido de nodos, haciéndolo adecuado para blockchains de consorcio con permisos [9].
En comparación con otros protocolos de consenso como Prueba de Trabajo (PoW) y Prueba de Participación (PoS), IBFT 2.0 ofrece finalidad inmediata, asegurando que una vez que una transacción se incluye en un bloque, no se puede alterar ni eliminar. La robustez de IBFT 2.0 se define en función de propiedades como la persistencia y la viabilidad, garantizando un registro de transacciones distribuido seguro y confiable. El protocolo opera en un modelo de red eventualmente sincrónica, asegurando la terminación determinista para el subprotocolo responsable de decidir sobre los bloques que se agregarán a la cadena de bloques. Además, IBFT 2.0 permite un conjunto dinámico de validadores, lo que permite agregar o eliminar nodos mediante un mecanismo de votación, a diferencia de los protocolos de consenso tradicionales con un conjunto fijo de nodos [1].

Smart Contracts

Un contrato inteligente es un contenedor de código que encapsula y replican los términos de los contratos del mundo real en el mundo digital, un contrato es un acuerdo entre dos partes o más en los que cada parte se compromete a cumplir con su parte del contrato, en el mundo real estos contratos son válidos ante la ley o algún órgano jurídico, sin embargo los Smart contracts sustituyen estos terceros de confianza o mediadores, para ello cuenta con la ejecución del código que se difunde y comprueba automáticamente por los nodos de la red blockchain [10].

Dentro de la EVM (Ethereum Virtual Machine) un contrato inteligente es un programa que se ejecuta dentro de la red Ethereum, es un conjunto de código y datos que existen en una dirección especifica de la red Blockchain. Un contrato inteligente es una especie de cuenta de Ethereum, es decir que tiene un equilibrio y puede ser objetivo de transacciones, no está controlado por un usuario si no que esta implementado en la red y se ejecuta según su programación [4].

UN contrato inteligente puede usarse como una aplicación independiente y también para implementar el backend de una aplicación, las aplicaciones que están basadas en contratos inteligentes son conocidas como DApps. Es decir que los contratos inteligentes son la lógica del negocio [11].

NFT

Los NFT son tokens no fungibles por sus siglas en inglés (Non-Fungible Token), es una forma única de un activo digital respaldado por la tecnología blockchain, a diferencia de otros activos como las criptomonedas un NFT representa la propiedad exclusiva e indivisible de un activo digital especifico, estos pueden ser por ejemplo obras de arte, videos, música, video juegos y otros contenidos digitales únicos [12].

En los últimos años tanto la investigación como las aplicaciones de NFT en otros campos ha crecido como en el campo de la educación donde algunas de las aplicaciones van desde la creación de tokens para los recursos de estudio, para la gestión de becas, depósitos y también para el registro de los estudiantes [13].

2 METODOLOGÍA

2.1 Metodología Iterativo Incremental

Dentro de esta sección, se detalla la metodología utilizada para el desarrollo del componente del prototipo. La metodología utilizada es la metodología de desarrollo Iterativo e incremental, la cual permite entregas de versiones parciales del prototipo, además de que dentro del desarrollo se toma en cuenta la retroalimentación del cliente. Además, esta metodología permite adaptar los requerimientos mientras el prototipo cambia en el tiempo, también permite la incorporación de cambios de manera más fácil y rápida [14].

A continuación, se detallan las fases del modelo iterativo incremental:

- Requerimientos: esta fase se refiere a los objetivos del prototipo [15].
- Definición de las tareas e iteraciones: tomando en cuenta los objetivos se hace una lista de tareas y agruparlas en iteraciones que tendrá el prototipo [15].
- Diseño de incrementos: Se define cual será la evolución del producto en cada iteración [15].
- Desarrollo del incremento: se realizan las tareas previstas y se desarrollan los incrementos establecidos [15].
- Validación de incrementos: cuando se acaba cada iteración, se debe revisar si se cumplió con lo previsto en el incremento y si es necesario volver hacia atrás [15].
- Integración de incrementos: luego de validar los incrementos, los incrementos dan forma a lo que se denomina línea incremental o evolución del proyecto, donde cada incremento ha contribuido al resultado final [15].
- Entrega del producto: cuando el producto en su conjunto es validado y se confirma que esté de acuerdo con los objetivos iniciales, entonces el proyecto se entrega [15].

2.2 Desarrollo

Definición de tareas e iteraciones

Dentro de esta fase se define cada iteración y el periodo de tiempo de cada una, las cuales serán desarrolladas cronológicamente.

A continuación, se muestra la Tabla 1 con el plan de las iteraciones planteados para el desarrollo del componente del prototipo.

Tabla 1 Iteraciones del desarrollo del prototipo

#	Iteración	Descripción	Fecha Inicio	Fecha Fin
1	Establecer los requerimientos para el componente.	Se crean las historias de usuario para el componente del prototipo. Establecer la estructura de los datos que se van a utilizar.	04/12/2023	24/12/2023
2	Crear la red Blockchain.	Se define la tecnología blockchain a usarse y se despliega la red blockchain.	25/12/2023	14/01/2024
3	Desarrollar la lógica del prototipo	Se crean los contratos inteligentes que le van a dar la lógica al componente del prototipo.	15/01/2024	28/01/2024
4	Desarrollar pruebas.	Se crean scripts de prueba, para probar la funcionalidad del componente	29/01/2024	11/02/2024

Diseño de incrementos

Dentro de esta fase se especifica el avance del prototipo en cada iteración.

Para la primera iteración se usarán historias de usuario para representar las funcionalidades o características que se espera obtener desde la perspectiva del usuario del componente del prototipo de la aplicación [16], el modelo de historia de usuario usado se encuentra en la Tabla 2 en el anexo.

Para la siguiente iteración se espera obtener una red blockchain Hyperledger Besu con cuatro nodos que se comuniquen entre sí y utilicen el protocolo de consenso IBFT 2.0.

Para la tercera iteración ya se cuenta con la red creada, por lo que se establece la lógica del prototipo mediante los contratos inteligentes, siguiendo las historias de usuario.

En la última iteración se tiene el prototipo listo y se realizan las pruebas para verificar que se cumple con los requisitos establecidos en las historias de usuario.

Desarrollo de incrementos

En esta fase se realiza el diseño de cada iteración de acuerdo con el plan establecido para el componente del prototipo.

- **Primer Incremento**

  Se definen las historias de usuario para el componente del prototipo de sistema de identificación estudiantil.

Tabla 2 Historia de Usuario – Emitir Identidad

Historia de Usuario	
N.º 1	Usuario: Administrador
Nombre de la historia: Emitir Identidad	
Prioridad de negocio: Alta	Riesgo de Desarrollo: Alta
Programadores responsables: Sergio Naranjo	Iteración Asignada: 3
Descripción: El administrador debe ser capaz de emitir una credencial para un estudiante	
Observaciones: Ninguna	

Tabla 3 Historia de Usuario – Verificar Identidad

Historia de Usuario	
N.º 2	Usuario: Verificador
Nombre de la historia: Verificador	
Prioridad de negocio: Alta	Riesgo de Desarrollo: Alta
Programadores responsables: Sergio Naranjo	Iteración Asignada: 3
Descripción: El verificador debe ser capaz de verificar si una credencial es válida para el usuario que la presenta y si está dentro del rango de fechas en el que es válido.	
Observaciones: Ninguna	

Tabla 4 Historia de Usuario - Revocar Identidad

Historia de Usuario	
N.º 3	Usuario: Administrador
Nombre de la historia: Revocar Identidad	
Prioridad de negocio: Alta	Riesgo de Desarrollo: Alta
Programadores responsables: Sergio Naranjo	Iteración Asignada: 3
Descripción: El administrador debe ser capaz de revocar una credencial emitida, agregando un dato a la blockchain.	
Observaciones: Ninguna	

Para determinar los datos que son necesarios dentro de una credencial que se emite a un estudiante se consultó con diferentes expertos, y se llegó a la conclusión de que los datos que se deben incluir son los que se muestran en la Tabla 5.

Tabla 5 Datos utilizados en la credencial

Identificador	Descripción	Tipo
id	Identificador único	number
nombres	Nombres del estudiante	string

apellidos	Apellidos del estudiante	string
facultad	Facultad a la que pertenece	string
semestre	Semestre académico actual	number
universidad	Universidad del estudiante	string
fechaEmision	Fecha en la que es creado la identificación	number
fechaExpiracion	Fecha hasta la cual la identidad emitida es valida	number

Para manejar los datos cuando una identificación es revocada se crea un registro en la blockchain donde se ingresan los siguientes datos.

Tabla 6 Datos de credencial revocada en la blockchain

Identificador	Descripción	Tipo
tokenID	Identificador único de la credencial en la blockchain	number
fechaDeBaja	Fecha en la cual se dio de baja la credencial	number

- **Segundo Incremento**

Para el segundo incremento, se procede a definir la plataforma blockchain que será utilizada. Después de analizar las diversas opciones para implementar la red blockchain y considerar las necesidades específicas del prototipo, tales como que permita la creación de una red privada, que posibilite la modificación de los protocolos de consenso y permita la ejecución de contratos inteligentes. Se ha concluido que la elección mas adecuada es Hyperledger Besu. Este cliente Ethereum ofrece la capacidad de crear redes privadas y dispone de

varios algoritmos de consenso. Asimismo, posibilita la ejecución de contratos inteligentes escritos en Solidity, según la información recabada en [1].

Para desplegar la red se usó una instancia EC2 con las siguientes características:

Tabla 7 Características de la Instancia EC2

CPU	(t3.large) 2vCPU x86_64 Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz
RAM	8GB
Almacenamiento	30GB
Red	http://52.7.204.248/

La instancia EC2, está configurada con una IP elástica “52.7.204.248”. Por medio del grupo de seguridad de la instancia, se implementó una limitación de acceso para que solo se tenga acceso desde la IP desde donde se está desarrollando el prototipo.

Para desplegar la red, se siguió la documentación oficial de Hyperledger Besu, donde se detalla paso a paso la creación de una red privada Besu [17]. Se hace uso de la Quorum Developer Quickstart que utiliza la imagen de Hyperledger Besu en Docker y ayuda a la creación de una red blockchain privada [18].

Para empezar la creación de la red se necesita de las herramientas: Docker, Docker Compose, Node.js v12 o superior y Hardhat.

Para inicializar la red se usa el comando “npx quorum-dev-quickstart” que permite crear los archivos y artefactos que Docker-compose necesita, (ver Figura 3).

Figura 3 Creación de archivos necesarios para la red

A continuación, se selecciona la opcion Hyperledger Besu, luego se selecciona otras opciones y al finalizar se obtiene el directorio “quorum-test-network” (ver Figura 4).

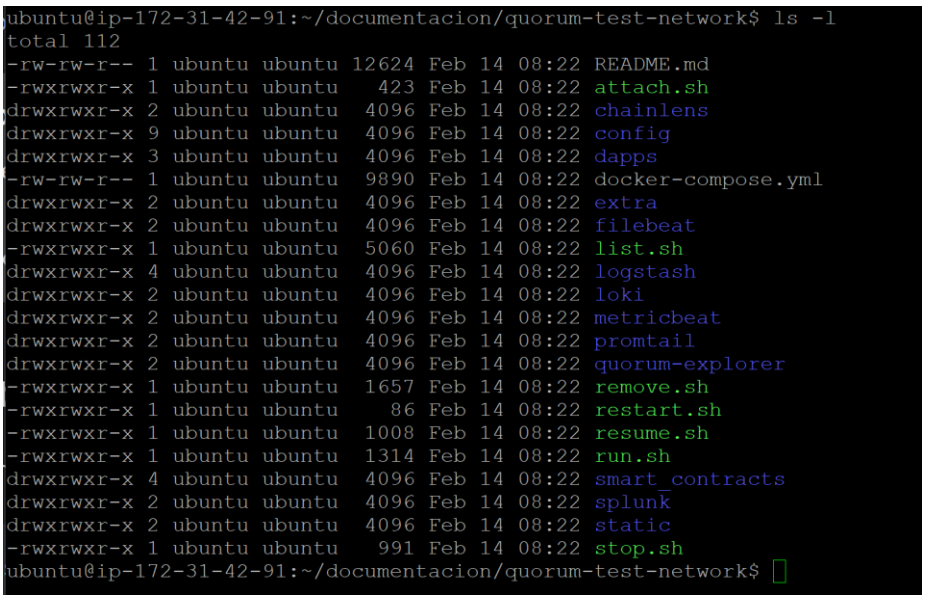


Figura 4 Directorio de la red Hyperledger Besu

Uno de los archivos principales en el directorio es “ibftConfigFile.json” que es el archivo génesis de la red, en este archivo se define el protocolo de consenso, el número de nodos que se va a usar en la red (ver Figura 5).

Figura 5 Archivo génesis de la red

Además, en el archivo “ibftConfigFile.json” también se definen las cuentas que están permitidas dentro de la red (ver Figura 6). Para este prototipo solo se activaros tres cuentas dentro de la red blockchain.

Figura 6 Cuentas permitidas dentro de la red

Con estas especificaciones podemos inicializar la red con el comando “./run.sh” (ver Figura 7). Se puede ver como se ha creado la red y se muestra además una lista de los servicios que se han creado (ver Figura 8).

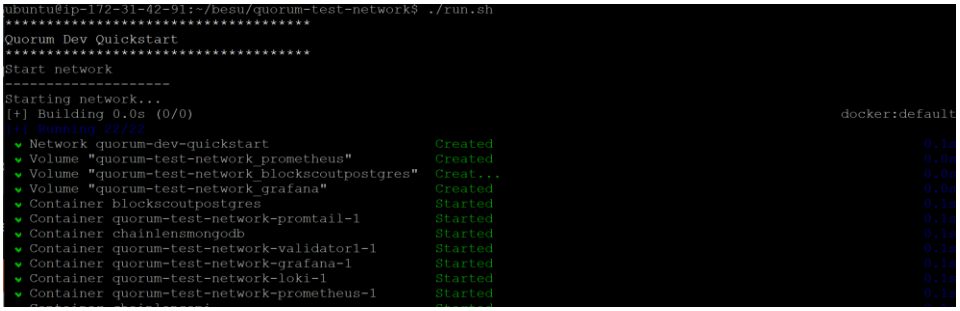


Figura 7 Inicio de la red

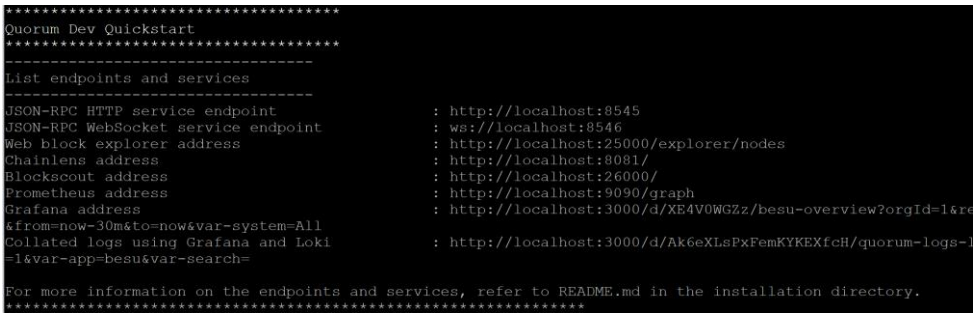


Figura 8 Servicio web creados

En [18], se encuentra más información sobre mas comandos de gestión de la red como por ejemplo como terminar la red, como remover la red o reiniciar la red.

Para inicializar el servicio de Chainlens, hay que dirigirse a un navegador a la dirección “http://52.7.204.248:8081/” (ver Figura 9), donde se muestra la interfaz gráfica para poder explorar la red.

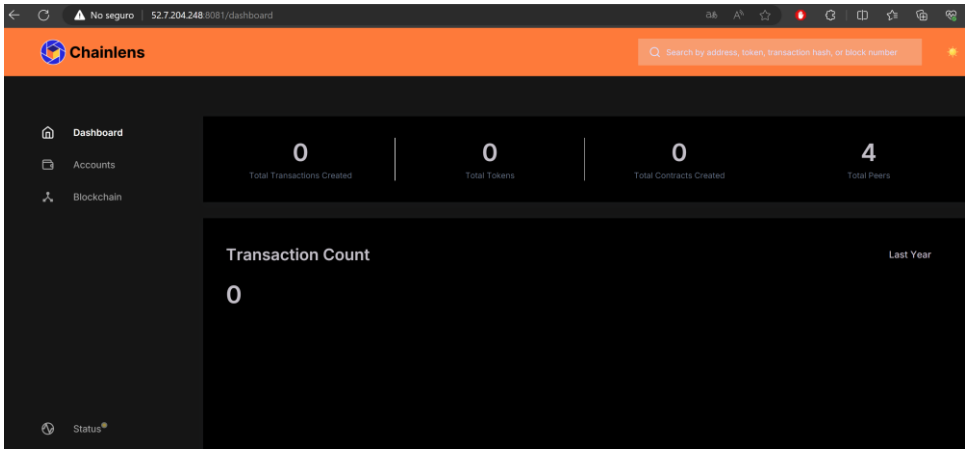


Figura 9 Servicio Chainlens

- **Tercer Incremento**

  Para el tercer incremento en primer lugar se tiene que configurar el ambiente de desarrollo, para ello se usó Visual Studio Code conectado a una máquina virtual WSL, además de instalar Node.js. Para el desarrollo de los contratos inteligentes se hace uso de Hardhat, un framework de Ethereum que se utiliza principalmente para la creación, prueba e implementación de contratos inteligentes en Ethereum.

  ### Configuración del ambiente de desarrollo

  Para instalar Hardhat, se debe inicializar un nuevo proyecto con Node.js y luego usar el comando “npm install --save-dev hardhat@latest”, Una vez instalado se usa el comando “npx hardhat init” para iniciar Hardhat (ver Figura 10).

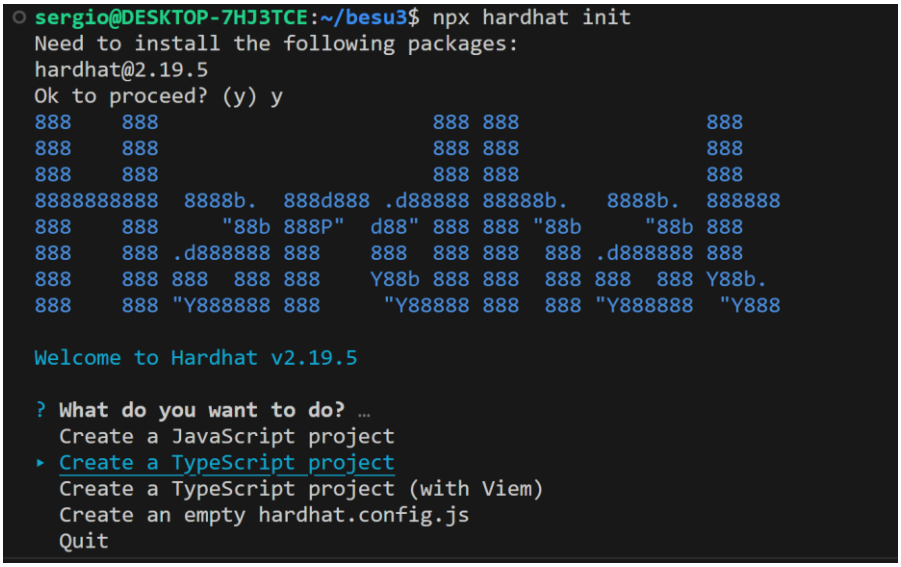


Figura 10 Inicialización de HardHat

  Se selecciona la opcion de crear un proyecto TypeScript y continuar con la instalación, una vez culminada la instalación se obtiene un directorio (ver Figura 11), donde el directorio “Contracts” contendrá los contratos inteligentes del prototipo, el directorio “Scripts” contendrá los scripts tanto para desplegar los contratos en la red como para probarlos.

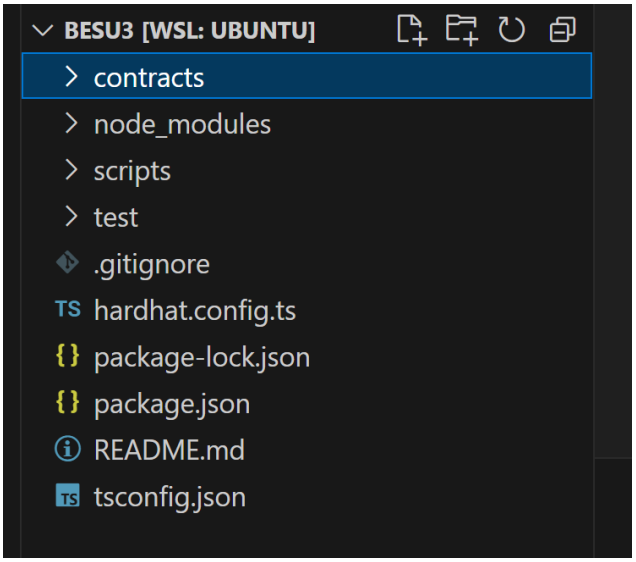


Figura 11 Directorio del proyecto

El archivo “hardhat.config.ts” se usa para configurar la red que se va a usar para el desarrollo de los contratos, el archivo de configuración contiene la configuración para la conexión con la red blockchain [19] (ver Figura 12).

```
module.exports = {
  defaultNetwork: "hardhat",
  networks: {
    // in built test network to use when developing contracts
    hardhat: {
      chainId: 1337
    },
    besu: {
      url: "http://52.7.204.248:8545",
      chainId: 1337,
      // test accounts only, all good ;)
      accounts: [
        "0x8f2a55949038a9610f50fb23b5883af3b4ecb3c3bb792cbcefbd1542c692be63",
        "0xc87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3",
        "0xae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f"
      ]
    }
  },
  solidity: {
    version: "0.8.20",
    settings: {
      optimizer: {
        enabled: true,
        runs: 200
      }
    }
  },
```

Figura 12 Configuración de la red Hardhat

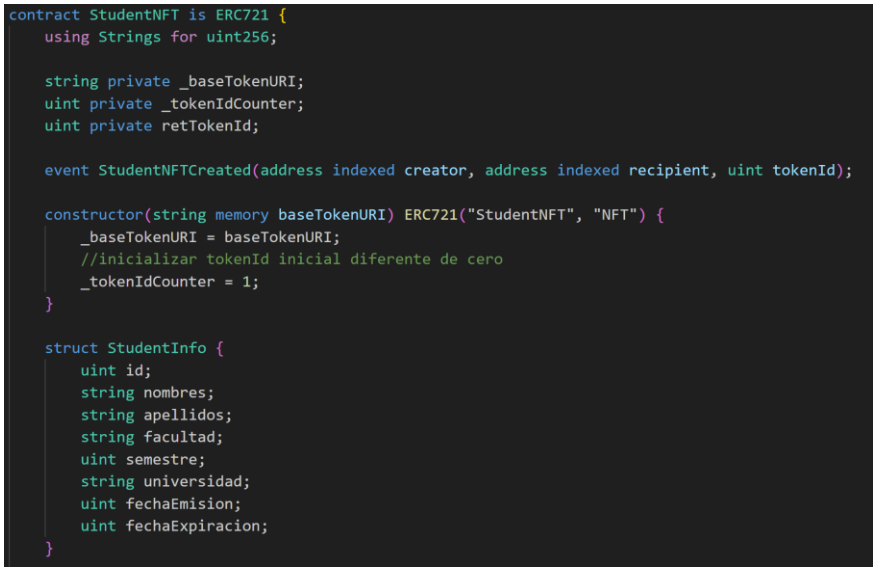
Desarrollo de los contratos inteligentes

Emisión de credenciales.

El primer requisito planteado en la primera fase es la emisión de una identidad para un usuario, para este fin se pensó en el uso de NFT ya que con esta tecnología se puede garantizar la autenticidad de las credenciales, permiten un acceso más rápido y un control directo sobre la identidad [13].

Para crear un NFT es necesario instalar la dependencia “npm install @openzeppelin/contracts”, además es necesario seguir la documentación del estándar ERC-721 para la creación de Tokens No Fungibles (NFT) [20].

La Figura 13, muestra la declaración del contrato inteligente, junto con la creación de la estructura de datos que tendrá el.



```
contract StudentNFT is ERC721 {
    using Strings for uint256;

    string private _baseTokenURI;
    uint private _tokenIdCounter;
    uint private retTokenId;

    event StudentNFTCreated(address indexed creator, address indexed recipient, uint tokenId);

    constructor(string memory baseTokenURI) ERC721("StudentNFT", "NFT") {
        _baseTokenURI = baseTokenURI;
        //inicializar tokenId inicial diferente de cero
        _tokenIdCounter = 1;
    }

    struct StudentInfo {
        uint id;
        string nombres;
        string apellidos;
        string facultad;
        uint semestre;
        string universidad;
        uint fechaEmision;
        uint fechaExpiracion;
    }
```

Figura 13 Declaración de StudentNFT

La función “generateTokenId” (ver Figura 14), genera un id para el NFT único a partir de los datos del estudiante y el número de contrato generado por el contrato inteligente.

```
//Generacion del TokenId
function generateTokenId(uint _id, string memory _nombres, string memory _apellidos, uint  _fechaEmision,
    uint256 hashResult = uint256((keccak256(abi.encodePacked(_id, _nombres, _apellidos, _fechaEmision, _f
    hashResult =uint256((keccak256(abi.encodePacked(_tokenIdCounter, _id))));
    return hashResult;
}
```

Figura 14 Función generateTokenId

La función “createStudentNFT” (ver Figura 15), se en carga de crear el NFT con la información del estudiante, además de asignarle el NFT al destinatario.

```
function createStudentNFT(
    uint _id,
    string memory _nombres,
    string memory _apellidos,
    string memory _facultad,
    uint _semestre,
    string memory _universidad,
    uint  _fechaEmision,
    uint  _fechaExpiracion,
    address recipient // direccion del destinatario
) public {
    _tokenIdCounter ++;
    uint tokenId = generateTokenId(_id, _nombres, _apellidos
    mintStudentNFTTo(recipient, tokenId, _id, _nombres, _ape
    retTokenId = tokenId;
    emit StudentNFTCreated(msg.sender, recipient, tokenId);

}
```

Figura 15 Función createStudentNFT

Finalmente, las funciones “getStudentInfo” y “getTokenId” (ver Figura 16), sirven para devolver la información del usuario y el “tokenId” del NFT creado.

```
function getStudentInfo(uint tokenId) public view returns (StudentInfo memory) {
    return _studentInfo[tokenId];
}

function getTokenId() public view returns (uint){
    return retTokenId;
}
```

Figura 16 Funciones getStudentInfo y getTokenId

Revocación de credenciales

Dado que los datos en la blockchain son inmutables, lo que quiere decir que no es factible colocar un campo para revocación dentro de una credencial NFT, se implementa un contrato inteligente que registre en la blockchain el “tokenId” y la “fecha” cuando un NFT es revocado.

La Figura 17, muestra la declaración del contrato inteligente “RevocNFTStudent” además de la estructura de datos que se enviara a la blockchain.

```
contract RevocNFTStudent {
    mapping(uint => DatosNFTEstudiante) private _datosNFT;

    event NFTDadoDeBaja(address indexed deregistrador, uint indexed tokenId);

    struct DatosNFTEstudiante {
        uint tokenId;
        address deregistrador;
        uint fechaDeBaja;
    }
```

Figura 17 Declaración RevocNFTStudent

Las funciones “darDeBajaNFT” y “verificarNFTDadoDeBaja” (ver Figura 18), permiten dar de baja a una credencial NFT si no ha sido revocada antes.

```
function darDeBajaNFT(uint tokenId) public {
    if (verificarNFTDadoDeBaja(tokenId)) {
        revert("NFT ya dado de baja");
    }
    _datosNFT[tokenId].fechaDeBaja = block.timestamp;
    emit NFTDadoDeBaja(msg.sender, tokenId);
}

function verificarNFTDadoDeBaja(uint tokenId) public view returns (bool) {
    return _datosNFT[tokenId].fechaDeBaja != 0;
}
```

Figura 18 Funciones “darDeBajaNFT” y “verificarNFTDadoDeBaja”,

Verificar credenciales

Para verificar a un estudiante se crea un contrato inteligente, que tiene que importar los dos contratos creados anteriormente (ver Figura 19), ya que debe tener acceso a funciones definidas en ambos contratos.

```
import "@openzeppelin/contracts/token/ERC721/IERC721.sol";
import "./NFT_Student.sol"; // Import the StudentNFT contract;
import "./NFT_Revoc.sol"; //importar el validador de revocacion

// Definicion del verificador
contract StudentNFTVerifier {
    StudentNFT private studentNFTContract;
    RevocNFTStudent private revocNFTContract;

    constructor(address studentNFTAddress, address revocNFTAddress) {
        studentNFTContract = StudentNFT(studentNFTAddress);
        revocNFTContract = RevocNFTStudent(revocNFTAddress);
    }
```

Figura 19 Declaracion StudentNFTVerifier

La función “validateDateRevoc” (ver Figura 20), toma como parámetros la fecha de expiración del NFT y el “tokenId” del NFT para verificar si no ha caducado el NFT y si no ha sido revocado.

Los datos de fechas son manejados como enteros como unix timestamp.

```
function validateDateRevoc(uint fechaExpiracion, uint tokenId) public view returns (bool) {
    // Comparar con la fecha actual
    if (fechaExpiracion > block.timestamp && !revocNFTContract.verificarNFTDadoDeBaja(tokenId)) {
        return true;
    }else {
        return false;
    }
}
```

Figura 20 Función validateDateRevoc

Finalmente, la función “validateNFT” y “getStudentInfo”, (ver Figura 21), sirven para retornar si un NFT es válido y la información del NFT para validar en la aplicación.

```
function validateNFT(uint tokenId) public view returns (bool) {
    StudentNFT.StudentInfo memory studentInfo = studentNFTContract.getStudentInfo(tokenId);
    return validateDateRevoc(studentInfo.fechaExpiracion, tokenId);
}

function getStudentInfo(uint tokenId) public view returns (StudentNFT.StudentInfo memory) {
    return studentNFTContract.getStudentInfo(tokenId);
}
```

Figura 21 Funciones “validateNFT” y “getStudentInfo

Desplegar Contratos Inteligentes

Para desplegar los contratos inteligentes creados, se tiene que compilar los contratos con el comando “npx hardhat compile”.

Para el script de despliegue de los contratos inteligentes se debe tomar en cuenta que se deben desplegar primero los contratos de emisión de credenciales y el de revocación de credenciales, y al final el de validación ya que este hace referencia a los anteriores (ver Figura 22).

```
import { ethers } from "hardhat";

async function main() {
  const [deployer] = await ethers.getSigners();

  console.log('Deploying contracts with the account:', deployer.address);

  //Deploy NFT Creator
  const Contract = await ethers.getContractFactory('StudentNFT');
  const contract = await Contract.deploy('identificacion');

  console.log('Contract address:', contract.target);

   //Deploy NFT Revoc contract
   const NFTRevoc = await ethers.getContractFactory('RevocNFTStudent');
   const nftRevocContract = await NFTRevoc.deploy();

   console.log('NFTRevoc contract address:', nftRevocContract.target);

  // Deploy NFTVerifier contract
   const NFTVerifier = await ethers.getContractFactory('StudentNFTVerifier');
   const nftVerifierContract = await NFTVerifier.deploy(contract.target, nftRevocContract.target);

   console.log('NFTVerifier contract address:', nftVerifierContract.target);


}

main()
  .then(() => process.exit(0))
  .catch(error => {
    console.error(error);
    process.exit(1);
  });
```

Figura 22 Script para desplegar contratos inteligentes

Para correr el script de despliegue se necesita el comando “hardhat run” además de especificar a la red a la que se va a desplegar los contratos como se muestra en la Figura 23.

```
sergio@DESKTOP-7HJ3TCE:~/besu2$ npx hardhat run scripts/deploy.ts --network besu
Compiled 2 Solidity files successfully (evm target: paris).
Deploying contracts with the account: 0xFE3B557E8Fb62b89F4916B721be55cEb828dBd73
Contract address: 0x42699A7612A82f1d9C36148af9C77354759b210b
NFTRevoc contract address: 0xa50a51c09a5c451C52BB714527E1974b686D8e77
NFTVerifier contract address: 0x9a3DBCa554e9f6b9257aAa24010DA8377C57c17e
```

Figura 23 Contratos Inteligentes desplegados

- **Cuarto Incremento**

  Dentro del último incremento se crea tres scripts para probar la funcionalidad de los contratos inteligentes, cada uno toma un parámetro con la dirección del contrato inteligente en la blockchain.

  El primer script interactúa con la creación de una credencial NFT (ver Figura 24).

```
async function main() {
  const [deployer] = await ethers.getSigners();

  // Obtener la instancia del contrato
  const contractAddress = '0xfE0B7EE21e8298fC68b9Bf5f404e7df7B6671EC2';
  const studentNFTContract = (await ethers.getContractAt(
    'StudentNFT',
    contractAddress,
    deployer
  )) as StudentNFT;

  await studentNFTContract.createStudentNFT(
    201810929,
    'Juan Pedro',
    'Pablo Sanchez',
    'Facultad de Sistemas',
    5,
    'EPN - FIS',
    1707930000,
    1739552400,
    '0xFE3B557E8Fb62b89F4916B721be55cEb828dBd73'
  );
  // Obtener información de un estudiante por tokenId
  const tokenId: bigint = await studentNFTContract.getTokenId(); // Reemplaza con el tokenId real
  const studentInfo = await studentNFTContract.getStudentInfo(tokenId);
  console.log('Id del token creado: ', tokenId);
  console.log('Información del Estudiante:', studentInfo);
}
```

Figura 24 Script creación de credencial

El resultado del script es la creación de una credencial como se puede notar en la Figura 25.

```
sergio@DESKTOP-7HJ3TCE:~/besu2$ npx hardhat run scripts/crearNFT.ts --network besu
Id del token creado:  135194734970659703502074222091025426927324944183407123137441454840991629503 2n
Información del Estudiante: Result(8) [
  201810929n,
  'Juan Pedro',
  'Pablo Sanchez',
  'Facultad de Sistemas',
  5n,
  'EPN - FIS',
  1707930000n,
  1739552400n
]
sergio@DESKTOP-7HJ3TCE:~/besu2$
```

Figura 25 Ejecución del primer script

El segundo interactúa con la revocación de una credencial (ver Figura 26), este script también toma como parámetro el “tokenId” de la credencial.

```
async function main() {
  const [deployer] = await ethers.getSigners();

  // Dirección del contrato RevocNFTStudent.
  const revocNFTStudentAddress = "0xdD2777Be7749DE8cD44b67575557f6D3c2c0cF51";
  // TokenId del NFT
  const tokenId: bigint = 135194734970659703502074222091025426927324944183407123137441454840991629

  const revocNFTStudent = await ethers.getContractAt("RevocNFTStudent", revocNFTStudentAddress);

  // Verificar si el NFT ya ha sido dado de baja
  const yaDadoDeBaja = await revocNFTStudent.verificarNFTDadoDeBaja(tokenId);

  if (yaDadoDeBaja) {
    console.log(`El NFT con tokenId ${tokenId} ya ha sido dado de baja anteriormente.`);
  } else {
    // Dar de baja el NFT
    await revocNFTStudent.darDeBajaNFT(tokenId);
    console.log(`El NFT con tokenId ${tokenId} ha sido dado de baja con éxito.`);

    // Verificar nuevamente después de dar de baja
    const ahoraDadoDeBaja = await revocNFTStudent.verificarNFTDadoDeBaja(tokenId);
    console.log(`Verificación después de dar de baja: ${ahoraDadoDeBaja}`);
  }
}
```

Figura 26 Script dar de baja una credencial

El resultado de la ejecución del segundo script se muestra en la Figura 27, al segundo script se le pasa como parámetro el “tokenId” de la credencial generada con el script anterior.

```
sergio@DESKTOP-7HJ3TCE:~/besu2$ npx hardhat run scripts/RevocarNFT.ts --network besu
El NFT con tokenId 135194734970659703502074222091025426927324944183407123137441454840
Verificación después de dar de baja: false
sergio@DESKTOP-7HJ3TCE:~/besu2$
```

Figura 27 Ejecución del segundo script

Finalmente tenemos al script de verificación del NFT (ver Figura 28), este también toma como parámetro el “tokenId” del NFT.

```
async function main() {
  const [deployer] = await ethers.getSigners();

  //Direccion del contrato de verificacion.
  const StudentNFTVerifierAddress = "0xA86EB77c09aE0F2164065aB14094565011b0BfcA";
  //TokenId del NFT
  const tokenId: bigint = 135194734970659703502074222091025426927324944183407123137441454840991629

  const StudentNFTVerifier = await ethers.getContractAt("StudentNFTVerifier", StudentNFTVerifierAd
  const studentInfo = await StudentNFTVerifier.getStudentInfo(tokenId);
  const isValid = await StudentNFTVerifier.validateNFT(tokenId);

  if (isValid) {
    console.log("Es un NFT válido");
    console.log("Información del estudiante:", studentInfo);
  } else {
    console.log("Es un NFT inválido por la fecha de expiracion o por que ha sido dado de baja");
  }
}
```

Figura 28 Script verificación de NFT

El resultado de la ejecución del tercer script (ver Figura 29), muestra la verificación negativa de la credencial creado con el primer script.

```
sergio@DESKTOP-7HJ3TCE:~/besu2$ npx hardhat run scripts/verificarNFT.ts --network besu
Es un NFT inválido por la fecha de expiracion o por que ha sido dado de baja
sergio@DESKTOP-7HJ3TCE:~/besu2$
```

Figura 29 Ejecución del tercer script

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

Funcionamiento de la Red de Blockchain

Para comprobar que la red blockchain está funcionando correctamente se puede acceder a “http://52.7.204.248:8081/dashboard”, donde se puede obtener la información de la red Hyperledger Besu desplegada (ver Figura 30).

Figura 30 Chainlens red desplegada

Los Contratos Inteligentes se pueden ver desplegados en la red, en el apartado de Contracts (ver Figura 31), en Chainlens.

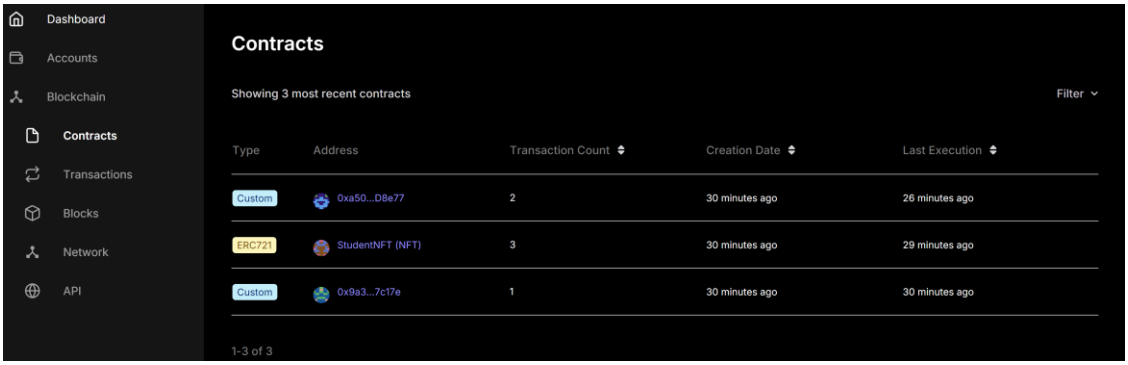


Figura 31 Smart Contracts desplegados en la red

Tiempo de respuesta

Con el objetivo de evaluar el tiempo de respuesta de la red blockchain Hyperledger Besu durante la ejecución de transacciones, se ha desarrollado un script específico (ver Figura 32) para medir el tiempo de respuesta de la red al validar un carnet. Este script se ejecutará 20 veces para obtener 20 mediciones consecutivas. Cada medición implica el registro de una transacción, en la blockchain, y se documenta el tiempo que la red demora en confirmar y validar esta transacción. Posteriormente, se determina el tiempo de respuesta promedio calculando la media de estos tiempos, proporcionando así un indicador representativo del rendimiento de la blockchain.

```
async function main() {
    const [deployer] = await ethers.getSigners();
    //Direccion del contrato de verificacion.
    const StudentNFTVerifierAddress = "0x9a3DBCa554e9f6b9257aAa24010DA8377C
    //TokenId del NFT
    const tokenId: bigint = 10246282135585013507091372587065766550785652811
    const StudentNFTVerifier = await ethers.getContractAt("StudentNFTVerifie

    const startTime = Date.now();
    const isValid = await StudentNFTVerifier.validateNFT(tokenId);
    // Registrar el tiempo de finalización
    const endTime = Date.now();
    // Calcular la duración total
    const duration = endTime - startTime;
    console.log('Transacción completada en', duration, 'milisegundos');
}

main()
    .then(() => process.exit(0))
    .catch((error) => {
        console.error(error);
        process.exit(1);
    });
```

Figura 32 Script para calcular el tiempo de respuesta

Con base a las mediciones realizadas, se ha determinado que el tiempo promedio de respuesta de una transacción es de 656.8 milisegundos. Este valor representa la duración desde el momento en que se inicia la transacción hasta que la red blockchain completa la validación y confirmación.

Rendimiento

Para evaluar el rendimiento de la red blockchain Hyperledger Besu se hará uso del servicio Grafana, que se despliega junto con la red de prueba creada en el puerto 3000, este servicio recoge los niveles de utilización tanto de CPU como de memoria RAM. Para las mediciones se toma como referencia la hora en la que se inicia la red blockchain.

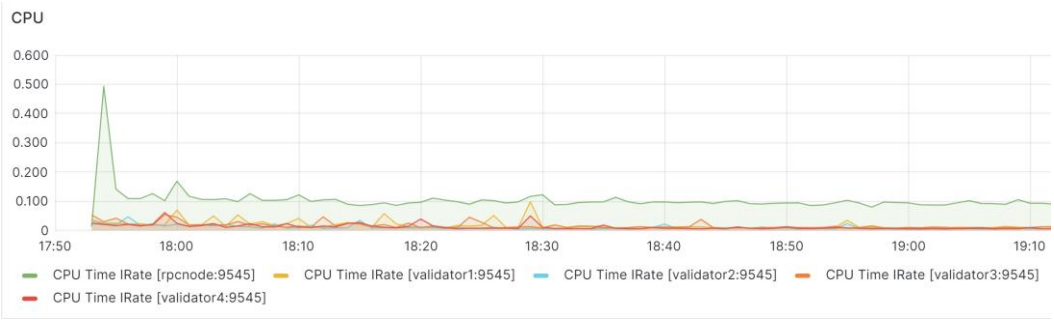


Figura 33 Rendimiento del CPU

Como se puede observar en la Figura 33, existe un pico del 50 % de uso de CPU al iniciar la red blockchain y después se mantuvo estable en alrededor del 20 % de uso de CPU.

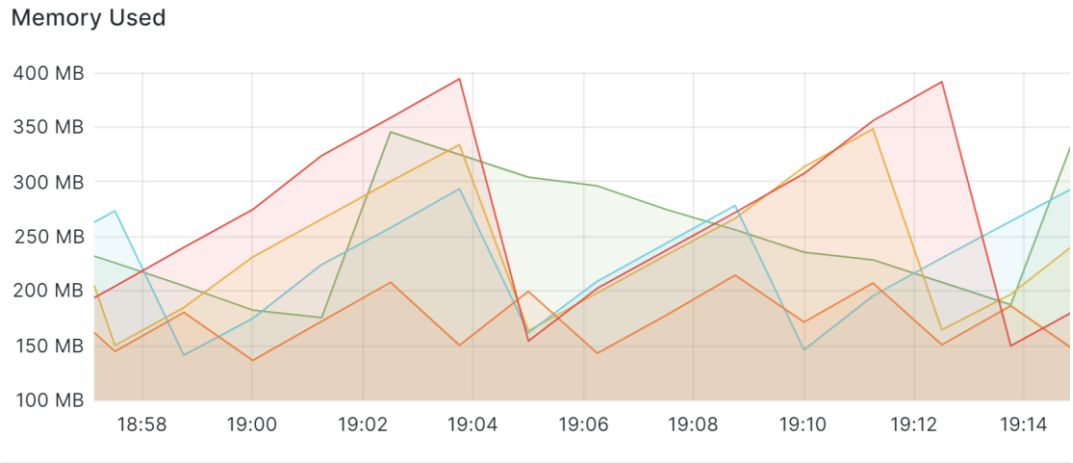


Figura 34 Uso de Memoria

En cuanto al uso de memoria se puede observar en la Figura 34, que el pico máximo de uso de memoria es de 400 MB.

3.2 Conclusiones

En conclusión, se logró crear el backend del prototipo de la aplicación, además los resultados obtenidos en la evaluación del rendimiento de la implementación de la blockchain para el sistema de identificación estudiantil son buenos. El tiempo de respuesta promedio de 656.8 milisegundos, existe un uso eficiente de la CPU con un pico inicial del 50% que se estabiliza luego en un 20%, y la gestión equitativa de la memoria, señalan una ejecución fluida y eficaz del sistema. Estos indicadores demuestran que nuestra implementación supera a otras soluciones típicas de blockchain en términos de eficiencia y optimización de recursos.

Es importante destacar que parte de estos resultados positivos se atribuyen al protocolo de consenso IBFT 2.0 empleado en nuestra blockchain. La elección de este protocolo ha demostrado ser acertada al contribuir significativamente a la reducción del tiempo de respuesta y a la gestión eficiente de los recursos de CPU y memoria.

La implementación de tecnologías de blockchain, como en este caso de Hyperledger Besu junto con NFTs y Smart Contracts, demuestra el potencial transformador que esta tecnología puede tener en diversos campos más allá de las criptomonedas. En particular, en el ámbito educativo, el uso de blockchain ofrece una solución segura, transparente y eficiente para la gestión de identificaciones estudiantiles.

La característica descentralizada de blockchain garantiza la integridad y la inmutabilidad de los datos, lo que es crucial en aplicaciones donde la verificación y la autenticidad son fundamentales. Los NFTs permiten la creación de identidades digitales únicas y no fungibles para los estudiantes, lo que mejora la seguridad y reduce el riesgo de fraudes.

3.3 Recomendaciones

Se recomienda antes de iniciar un proyecto analizar las diversas opciones de blockchain que hay disponible en el mercado actualmente, ya que puede parecer que las opciones para implementar blockchain son limitadas al inicio.

Para futuros trabajos, se recomienda recrear un ambiente real donde cada nodo se encuentre en una maquina diferente, además de agregar más nodos, para así de esta manera poder medir rendimiento y la escalabilidad que puede llegar a tener la red Hyperledger Besu.

También es recomendable experimentar con diferentes configuraciones de la red o con diferentes redes y protocolos de consenso para poder hacer una mejor elección de la tecnología que se va a utilizar.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] F. A. Pujol, H. Mora, T. Ramírez, C. Rocamora, and A. Bedón, "Blockchain-Based Framework for Traffic Event Verification in Smart Vehicles," *IEEE Access*, vol. 12, pp. 9251–9266, 2024, doi: 10.1109/ACCESS.2024.3352738.

[2] C. Fan, C. Lin, H. Khazaei, and P. Musilek, "Performance Analysis of Hyperledger Besu in Private Blockchain," in *Proceedings - 4th IEEE International Conference on Decentralized Applications and Infrastructures, DAPPS 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 64–73. doi: 10.1109/DAPPS55202.2022.00016.

[3] R.-V. Tkachuk, D. Ilie, R. Robert, V. Kebande, and K. Tutschku, "On the performance and scalability of consensus mechanisms in privacy-enabled decentralized renewable energy marketplace," *Annals of Telecommunications*, Sep. 2023, doi: 10.1007/s12243-023-00973-8.

[4] Ethereum, "Introducción a los contratos inteligentes." Accessed: Feb. 04, 2024. [Online]. Available: https://ethereum.org/es/developers/docs/smart-contracts

[5] L. Jia, X. Chen, L. Liu, X. Wang, K. Xiao, and G. Xu, "Blockchain data secure sharing protocol based on threshold Paillier algorithm," *High-Confidence Computing*, vol. 3, no. 4, p. 100132, Dec. 2023, doi: 10.1016/j.hcc.2023.100132.

[6] P. Baena-Luna and E. García-Río, "TECNOLOGÍA BLOCKCHAIN: DESAFÍOS PRESENTES Y FUTUROS EN SU APLICACIÓN," *Revista Conhecimento Online*, vol. 2, pp. 258–273, Aug. 2022, doi: 10.25112/rco.v2.2859.

[7] F. A. Sunny *et al.*, "A Systematic Review of Blockchain Applications," *IEEE Access*, vol. 10, pp. 59155–59177, 2022, doi: 10.1109/ACCESS.2022.3179690.

[8] Hyperledger Foundation, "Hyperledger Besu." Accessed: Feb. 04, 2024. [Online]. Available: https://wiki.hyperledger.org/display/BESU/Hyperledger+Besu

[9] R. Saltini and D. Hyland-Wood, "IBFT 2.0: A Safe and Live Variation of the IBFT Blockchain Consensus Protocol for Eventually Synchronous Networks," Sep. 2019, Accessed: Feb. 13, 2024. [Online]. Available: https://arxiv.org/abs/1909.10194v1

[10] H. Taherdoost, "Smart Contracts in Blockchain Technology: A Critical Review," *Information (Switzerland)*, vol. 14, no. 2. MDPI, Feb. 01, 2023. doi: 10.3390/info14020117.

[11] N. F. Samreen and M. H. Alalfi, "An empirical study on the complexity, security and maintainability of Ethereum-based decentralized applications (DApps)," *Blockchain: Research and Applications*, vol. 4, no. 2, p. 100120, Jun. 2023, doi: 10.1016/J.BCRA.2022.100120.

[12] Ethereum, "Non-fungible tokens (NFT)." Accessed: Feb. 05, 2024. [Online]. Available: https://ethereum.org/en/nft

[13] C. H. Wu and C. Y. Liu, "Educational Applications of Non-Fungible Token (NFT)," *Sustainability (Switzerland)*, vol. 15, no. 1, Jan. 2023, doi: 10.3390/su15010007.

[14] M. D. Lema Iza and J. R. Ortiz Bedoya, "Desarrollo de un sistema de gestión integrado utilizando software libre con el modelo iterativo incremental para llevar el control de los

procesos en la empresa software y hardware," 2016, Accessed: Feb. 05, 2024. [Online]. Available: http://localhost/handle/27000/3685

[15] OBS Business School, "Características y fases del modelo incremental." Accessed: Feb. 05, 2024. [Online]. Available: https://www.obsbusiness.school/blog/caracteristicas-y-fases-del-modelo-incremental

[16] undefined, undefined, D. E. L. Guevara, and N. B. L. S. Palomino, "Automatización de requisitos: Historias de usuario generadas a partir de un modelo orientado a objetivos basado en el framework i*," *Interfases*, no. 011, pp. 57–72, 2018, doi: 10.26439/INTERFASES2018.N011.2953.

[17] Hyperledger Besu, "Private networks." Accessed: Feb. 05, 2024. [Online]. Available: https://besu.hyperledger.org/private-networks

[18] Hyperledger Besu, "Private networks." Accessed: Feb. 13, 2024. [Online]. Available: https://besu.hyperledger.org/private-networks

[19] Hardhat, "Getting started with Hardhat - Ethereum development environment for professionals by Nomic Foundation." Accessed: Feb. 13, 2024. [Online]. Available: https://hardhat.org/hardhat-runner/docs/getting-started#quick-start

[20] OpenZeppelin, "ERC 721." Accessed: Feb. 13, 2024. [Online]. Available: https://docs.openzeppelin.com/contracts/2.x/api/token/erc721

5 ANEXOS

ANEXO I. Formatos

Tabla 8 Modelo de Historia de Usuario

Historia de Usuario	
N.º #	Usuario:
Nombre de la historia:	
Prioridad de negocio:	Riesgo de Desarrollo:
Programadores responsables:	iteración Asignada:
Descripción:	
Observaciones:	

ANEXO II. Enlaces

La programación realizada se encuentra en el enlace de GitHub: https://github.com/SergioNaranjoS/besu2