SOFTWARE DE FACTURACIÓN ELECTRÓNICA USANDO TECNOLOGÍA BLOCKCHAIN

SEBASTIAN GUEVARA SANCHEZ

FUNDACIÓN UNIVERSITARIA CATÓLICA LUMEN GENTIUM

FACULTAD DE CIENCIAS BÁSICAS E INGENIERÍA

PROGRAMA DE INGENIERÍA DE SISTEMAS

SANTIAGO DE CALI

2021

SOFTWARE DE FACTURACIÓN ELECTRÓNICA USANDO TECNOLOGÍA BLOCKCHAIN

SEBASTIAN GUEVARA SANCHEZ

Proyecto presentado para optar al título de Ingeniero de Sistemas

Asesor

Bryan Garcia

FUNDACIÓN UNIVERSITARIA CATÓLICA LUMEN GENTIUM

FACULTAD DE INGENIERÍAS

INGENIERÍA DE SISTEMAS

SANTIAGO DE CALI

2021

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

_____

_____

_____

Firma del jurado

_____

Firma del jurado

Cali, 15 de junio de 2021

DEDICATORIA

En el tiempo que escribí este documento, sentí una gran alegría por lo que representa esta investigación, un esfuerzo de muchos meses de trabajo y de aprendizaje que cierra una carrera llena de todo tipo de experiencias. Aprovecho este espacio para dedicarle unas palabras a mis familiares por brindarme un soporte para mi crecimiento intelectual y ser el origen de mi inspiración, a mis amigos y compañeros por ser una gran fuente de apoyo para continuar avanzando.

A la Fundación Universitaria Católica Lumen Gentium y a la facultad de Ingeniería por todo el esfuerzo e inversión realizado para mi formación y la de mis compañeros. Por otorgarnos la oportunidad de crecer personal y profesionalmente, por permitirme representar a la universidad en varios congresos de tesistas y especialmente por concederme la oportunidad de egresar de sus aulas para fortalecer mi futuro.

Que mi Dios los bendiga a todos y que obtengan el mayor de todos los éxitos.

Sebastián Guevara Sánchez

AGRADECIMIENTOS

Quiero agradecerles primero a mis familiares por el continuo apoyo incondicional, para superar todo los desafíos y obstáculos que a lo largo de mi carrera y de esta investigación han aparecido.

También quiero agradecerle a la Fundación Universitaria Católica Lumen Gentium que me dio la oportunidad de estudiar en sus aulas, adquirir nuevos conocimientos y plantearme nuevos debates, planes y metas. También agradezco a sus docentes que me orientaron en estos estudios con pensamiento crítico, investigativo, mucho conocimiento y profesionalismo.

Igualmente agradezco al ingeniero Bryan Garcia, director de mi investigación por su orientación, apoyo y supervisión en este trabajo de grado, brindándome su tiempo, espacio y sabiduría para lograr este trabajo de grado.

Sebastián Guevara Sánchez

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

Actualmente la facturación electrónica está presentando vulnerabilidades al momento de almacenar su información porque se hace de manera centralizada y en la mayoría de casos en bases de datos SQL las cuales pueden recibir diferentes tipos de ataques de intrusos. La industria 4.0 ha presentado diferentes soluciones para este tipo de vulnerabilidades, una de estas soluciones es el Blockchain que con su almacenamiento descentralizado y cadena de bloques de información fortalece la seguridad informática de estas bases de datos. En este proyecto se desarrolla un software para resolver la problemática de seguridad en la facturación electrónica usando tecnología Blockchain. La metodología propuesta para el desarrollo de este proyecto es un estudio experimental, se realizará una investigación documental exploratoria para encontrar datos existentes y lograr resultados lógicos, también se usará el proceso de desarrollo de software SCRUM el cual empezará por una revisión bibliográfica en la que se detectarán las características y herramientas usadas en la facturación electrónica con tecnología Blockchain, seguido de un levantamiento de requisitos, una implementación de software y algunas pruebas para validar el software propuesto y para determinar los límites y robustez del sistemas creado. Con lo anterior se logrará una implementación de desarrollo de software que usará Blockchain para mejorar la seguridad en un programa de facturación electrónica en Colombia.

Palabras Claves: Bases de datos, Blockchain, Facturación electrónica, Industria 4.0, SCRUM, Seguridad informática.

ABSTRACT

Currently electronic invoicing is presenting vulnerabilities when storing your information because it is done centrally and in most cases in SQL databases which can receive different types of intruder attacks, thanks to industry 4.0 different have been presented solutions for this type of vulnerability, one of these solutions is the Blockchain that with its decentralized storage and chain of information blocks strengthens the computer security of these databases. In this project, software is developed to solve the security problem in electronic invoicing using Blockchain technology. The proposed methodology for the development of this project is an experimental study, an exploratory documentary investigation will be carried out to find existing data and achieve logical results, the SCRUM software development process will also be used, which will begin with a bibliographic review in which They will detect the characteristics and tools used in electronic invoicing with Blockchain technology, followed by a survey of requirements, a software implementation and some tests to validate the proposed software and to determine the limits and robustness of the system created. With the above, a software development implementation will be achieved that will use Blockchain to improve security in an electronic invoicing program in Colombia.

Keywords : Databases, Blockchain, Electronic invoicing, Industry 4.0, SCRUM, Computer security.

INTRODUCCIÓN

El Blockchain es una tecnología que sirve para volver más seguro el almacenamiento de información mediante “bloques de datos” que están enlazados entre sí por medio de criptografía [16]. Las facturas electrónicas tienen los mismos impactos legales de las facturas expedidas en papel. El SHA es una forma de encriptar información que se ha ido desarrollando y mejorando desde el primer SHA que existió, que fue el SHA-0 y el SHA-256 siendo una de las formas de encriptar más modernas, cada actualización de esta forma de encriptar es más segura que la anterior [2].

La facturación electrónica en Colombia es controlada por las normativas que da la DIAN, luego de estudiar los procedimientos para mantener segura la facturación electrónica se puede afirmar que la mayoría de facturas electrónica se almacenan en bases de datos centralizadas lo cual es muy vulnerable, ya que alguien puede acceder a esta base de datos y editar sus cifras por medio de una intrusión maliciosa. Por otra parte, las antiguas facturas electrónicas están encriptadas con SHA-1 y este tipo de encriptación en la actualidad es muy antiguo y presenta varios problemas de seguridad [13].

La compañía China que tiene el nombre de Tencent Holding hace pública su primera factura electrónica con Blockchain, el 10 de agosto de 2018, y un año después ya tenía 5300 empresas registradas para laborar con esta factura electrónica, luego de transcurrir este año lograron emitir 6 millones de facturas electrónica con Blockchain [33]. La compañía que lleva el nombre de Kiwiz en Francia lleva la delantera en Europa mejorando la seguridad informática de las facturas electrónicas con la encriptación de su almacenamiento en cadena de bloques [19]. La compañía belga llamada Peppol comenzó a ofrecer sus servicios de facturación electrónica con Blockchain en el año 2019 empezando a competir en el mercado europeo de las facturas electrónicas aseguradas con cadena de bloques [4]. Otra empresa que inició su competencia en Europa en el área de la industria 4.0 especialmente en Blockchain para facturación electrónica es Docuten, mejorando la protección de la información de sus facturas electrónicas [10]. En Colombia el gobierno nacional está empezando a implementar la facturación electrónica con Blockchain para las Pymes, también llamadas micro y medianas empresas, en una colaboración con una empresa privada

que lleva buen tiempo trabajando con esta tecnología llamada GoSocket [27]. En la actualidad varias empresas públicas y privadas están en la labor de migrar las facturas electrónicas al almacenamiento de información con Blockchain sin encontrarse problemas de intrusión o evasión fiscal. Considerando lo anterior podemos generar la siguiente hipótesis: la seguridad de la información que está en la contabilidad de las empresas mejora implementando Blockchain en la facturación electrónica.

La intención de este trabajo es crear un Software que almacene la información que hay en las facturas electrónicas usando tecnología Blockchain con encriptación SHA-256 o superior.

1. PLANTEAMIENTO DEL PROBLEMA

La reglamentación de la facturación electrónica en Colombia está en el decreto 2242 de 2015 y el decreto 1625 de 2016, desde los años en los que salieron estos decretos se empezó a migrar las facturas en papel a facturas electrónicas en diferentes empresas pero esta nueva forma de manejar las facturas ha presentado varios problemas porque estas facturas electrónicas se almacenan en bases de datos centralizadas las cuales tienen muchas vulnerabilidades en la actualidad, en Colombia las empresas también suelen almacenar sus facturas electrónicas en bases de datos SQL y en la mayoría de casos directamente en la DIAN [27].

La DIAN tiene un sistema llamado CUFE (Código único de factura electrónica) que se encarga de identificar como única cada factura por medio de un texto alfanumérico, este código CUFE hasta hace unos meses estaba encriptado en SHA-1 (Creado en 1995 por la NSA) el cual se puede vulnerar por medio de un “ataque de colisión” que consiste en alterar el hash para compararlo con otros hash y así poder descifrar la información que va dentro de este, en la actualidad la DIAN cifra sus CUFE por medio de SHA-256 o SHA-358 pero no todas las empresas se han actualizado a esta encriptación. Con el Blockchain se logra tener una trazabilidad, monitoreo, también se reducen los costos en el proceso de almacenamiento de información y se elimina la evasión fiscal [13].

Tomando en consideración lo expresado anteriormente se eligió el almacenamiento de información con tecnología Blockchain porque logra disminuir los gastos por pérdida de facturas o por cantidad de personal y también disminuye la posibilidad de errores en la administración de facturas, teniendo en cuenta lo anterior con la cadena de bloques encriptadas se rescata la seguridad en el proceso de la facturación electrónica. El Blockchain logra un mejor monitoreo, trazabilidad, alcanza una mejor transparencia y elimina la evasión fiscal; todo lo anterior se puede obtener porque en el Blockchain cuando se cambia un dato de un bloque de información y este no coincide con los otros bloques esto quiere decir que hubo un intento de estafa o desorden en una transacción [9].

1.1 FORMULACIÓN DEL PROBLEMA

La facturación electrónica representa menos gastos que la facturación en papel, en el año 2019 en Colombia había 13000 compañías usando facturación electrónica, pero la ley dice que en el año 2020 el 70% y en el año 2021 el 90% de los costos y gastos de las empresas deben estar con facturación electrónica, lo anterior quiere decir que si una empresa no ha empezado a emitir este tipo de facturación tendrá que pagar más impuestos [20]. Las problemáticas más visibles en el almacenamiento de la facturación electrónica está en el almacenamiento centralizado y su encriptación que normalmente está en SHA-1 la cual está descontinuado, otra contra tiempo es que estas bases de datos están en SQL el cual puede ser vulnerado por diferentes tipos de ataques de intrusos usando metodologías de hacking. Considerando lo anterior este proyecto busca definir y solucionar dicho problema mediante la pregunta de investigación ¿Con qué implementación de desarrollo de software usando Blockchain se puede mejorar la seguridad en un programa de facturación electrónica en Colombia?

1.2 SISTEMATIZACIÓN DEL PROBLEMA

La pregunta genera que se plantea en la formulación del problema y se divide en 3 preguntas que son:

¿Qué características y herramientas, usa un sistema de facturación electrónica con tecnología Blockchain?

¿De acuerdo con los requisitos identificados en la pregunta anterior, cómo se diseña y se implementa un software que use tecnología Blockchain para un sistema de facturación electrónico?

¿Cómo se puede validar el funcionamiento del software propuesto?

1.3 JUSTIFICACIÓN

La inversión en infraestructura y la innovación, son mecanismos fundamentales del crecimiento y el desarrollo económico. Se presenta un crecimiento en la inversión de industrias y de las tecnologías de la información gracias a que la población mundial empieza a vivir en ciudades. Según el MinTIC (2018) la tecnología Blockchain es una

gran oportunidad para las empresas del sector TI, generando nuevos mercados, y de esta manera permite la disminución de costos internos en las empresas, también soluciona los problemas de seguridad, descentralización de la información. El 1% de las empresas en Colombia ha adoptado esta tecnología según el Observatorio de Economía Digital de MinTIC. La DIAN ha presentado el modelo de facturación electrónica para ser implementado en el año 2019, el cual por medio de el manejo de información digital busca soportar transacciones de ventas de bienes y/o servicios. Sin embargo, este tipo de procesos conllevan a problemas de seguridad como pérdida de confidencialidad por acceso no autorizado en el proceso de registro de la factura, pérdida de la integridad de la información ocasionado por cambios incorrectos en el software, fraude originado por la no asignación de código QR en la generación del archivo XML, entre otras. En países como China la tecnología Blockchain se implementó para facturación electrónica de transporte público, mientras que en Colombia empresas como Carvajal Tecnología y Servicios han adoptado la tecnología Blockchain para fortalecer la seguridad de sus servicios de facturación electrónica. Con este proyecto se aporta al conocimiento desarrollando e implementando aplicaciones con tecnología Blockchain para que trabaje conjuntamente con la facturación electrónica minimizando riesgos de seguridad tecnológica.

1.4 OBJETIVOS

1.4.1 GENERAL

Desarrollar un software para sistemas de facturación electrónica colombiana usando tecnología Blockchain

1.4.2 ESPECÍFICOS

- Identificar los requerimientos tecnológicos para implementar un sistema de facturación electrónica usando tecnología Blockchain.
- Desarrollar un software que use tecnología Blockchain para un sistema de facturación electrónico de acuerdo con los requisitos identificados.
- Validar el funcionamiento del software propuesto.

2. MARCO REFERENCIAL

En este proyecto de grado se detectaron cuáles son las características y herramientas más usadas en la facturación electrónica con tecnología Blockchain, también se hará un desarrollo de software en el que se analizará las características para realizar unos diseños de software y posteriormente implementar el código.

2.1 ANTECEDENTES

2.1.1 Antecedentes de facturación electrónica con tecnología Blockchain

En el mundo se han logrado implementar varios softwares de facturación electrónica con tecnología Blockchain usando variedad de herramientas para lograr objetivos en común, algunos de estos desarrollos de software se muestran en la Tabla 1 a continuación:

Tabla 1. Herramientas software de programación en facturación electrónica con Blockchain

Referencia	**Descripción**	**Lenguaje de programación usado**
GoSocket	Este software tiene una API creada con C# la cual procesa facturas electrónicas de micro y medianas empresas en Colombia usando tecnología Blockchain [27] [14].	C#
WeChat	Esta empresa administraba programas de comunicaciones, pero decide invertir en un software que crea un entorno para cumplir con todos los requisitos de una facturación electrónica, la verificación de contratos y el pago que se puede efectuar usando cadena de bloques [31].	C++
Fisco BCOS	Esta plataforma tiene un proyecto de código abierto en el que se manejan facturas electrónicas con el Blockchain en el cual	Solidity , Python

	colaboran más de diez mil desarrolladores de software y miles de instituciones trabajan en conjunto desde el 2017 [32] [12].	
Docuten	El software de Docuten usa las redes Blockchain en facturaciones electrónicas y esta combinación le permite contar con una garantía transnacional dotada de inmutabilidad, transparencia y descentralización, este software está disponible para diferentes empresas [24].	Solidity

2.1.2 Antecedentes de desarrollo de software con tecnología Blockchain

Tanto en Colombia como en el mundo el Blockchain ha tenido mucha acogida para mejorar los sistemas tanto de facturas electrónicas como de votación, contratos inteligentes, seguimiento de transporte, entre otros, a continuación, en la Tabla 2 se muestra unos ejemplos de software con Blockchain y sus herramientas:

Tabla 2. Lenguajes de programación usados en proyectos con Blockchain

Referencia	Descripción	Herramientas Usadas
Elecciones de personero con Blockchain en Colombia	En este proyecto de votaciones se usó el framework VueJs y el desarrollo integraba el plugin de MetaMask para el navegador, para la conexión del Front-end con el Back-end se usó una comunicación HTTP por servicios REST con la librería Axios, se instaló en los servidores de VIvelab de Ethereum, las pruebas se hicieron en TestRPC [29].	Javascript

Ethertalliacer	Este contrato inteligente usa uno de los lenguajes que más se utilizan en este tipo de contratos Solidity ya que este es un lenguaje orientado a objetos[6].	Solidity
Corda	Corda es una plataforma que genera contratos inteligentes y está creada con los lenguajes de programación Java, javascript y Kotlin, es un buen competidor para otras plataformas como Ripple, ya que las aplicaciones CorDapp usan poco código para lograr grandes lógicas [7] [5].	Javascript, Java, Kotlin
Circle	Es un software que funciona gestionando pagos inteligentes en los que se manejan transacciones e información importante de sus clientes, fue creado con el lenguaje javascript [8].	Javascript
Hyperledger	Hyperledger es una comunidad de código abierto centrada en desarrollar un conjunto de marcos, herramientas y bibliotecas estables para implementaciones de blockchain de nivel empresarial. Es una plataforma para varios libros de contabilidad distribuidos, incluidos Hyperledger Fabric, Sawtooth, Indy así como herramientas como Hyperledger Caliper, bibliotecas como Hyperledger Ursa y hyperledger composer y javascrip [15].	Javascript, Java

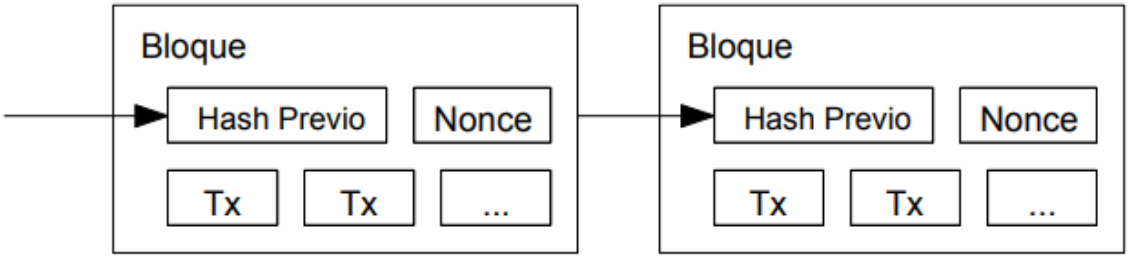
2.2 MARCO TEÓRICO

2.2.1 Blockchain

El Blockchain es una forma de procesar pagos electrónicos de manera segura basándose en pruebas criptográficas, este sistema le permite a dos partes interesadas realizar transacciones directamente sin la necesidad de un tercero confiable [30].

Luego de conocer las necesidades de los usuarios y la evolución de la tecnología, se han creado unas técnicas nuevas que son participativas, alcanzables y públicas. Estas nuevas tecnologías de la industria 4.0 se interesan en que las diversas empresas privadas y estatales entiendan la importancia de este tipo de tecnología y logren adquirir sus beneficios de acuerdo con el contexto de negocio [21]. A continuación, en la Figura 1 se muestra cómo se almacena el bloque anterior en el Hash previo.

Figura 1. Bloques con hash previo



Fuente: Bitcoin: Un Sistema de Efectivo Electrónico Usuario-a-Usuario [30]

2.2.1.1 Características de Blockchain

Las características que tiene el Blockchain son:

Transparencia: Cada registro o transacción que se realice tiene una copia de seguridad encriptada que lo respalda [22]. Si una entidad maliciosa intenta cambiar la información dentro del Blockchain tendrá que cambiar las copias de seguridad que están en los diferentes nodos. Otra característica que hace esta tecnología transparente es que los nodos no aceptan una transacción inválida [30].

Tecnología descentralizada: esta característica hace referencia a que no hay una entidad o persona que mande sobre los registros, esta propiedad ayuda a que no haya posibilidades de corrupción en las transacciones [22].

Seguridad mejorada: No hay una persona o entidad central de control. Todas las transacciones tienen una encriptación difícil de romper, lo anterior ayuda a que nadie pueda cambiar información para su beneficio propio [22]. Otra característica, que demuestra la seguridad mejorada de Blockchain, es que la encriptación de estas transacciones es computacionalmente poco posible de revertir, esto protege a los usuarios de fraude. Por otro lado, la marca de tiempo le comprueba al usuario que la información existió en el tiempo antes de ingresar al hash, cada marca de tiempo incluye la marca de tiempo anterior en su hash interno, formando así una cadena con las diferentes marcas de tiempo de las anteriores transacciones. De igual manera, un porcentaje de fraude es posible, puede ser hackeado por un gran poder de procesamiento [30].

Registros distribuidos: Las transacciones e información dentro de Blockchain son distribuidos con copias a nivel mundial, esto quiere decir que si alguien quiere corromperla red tendrá que realizar cambios en la información de diferentes lugares y al mismo tiempo [22].

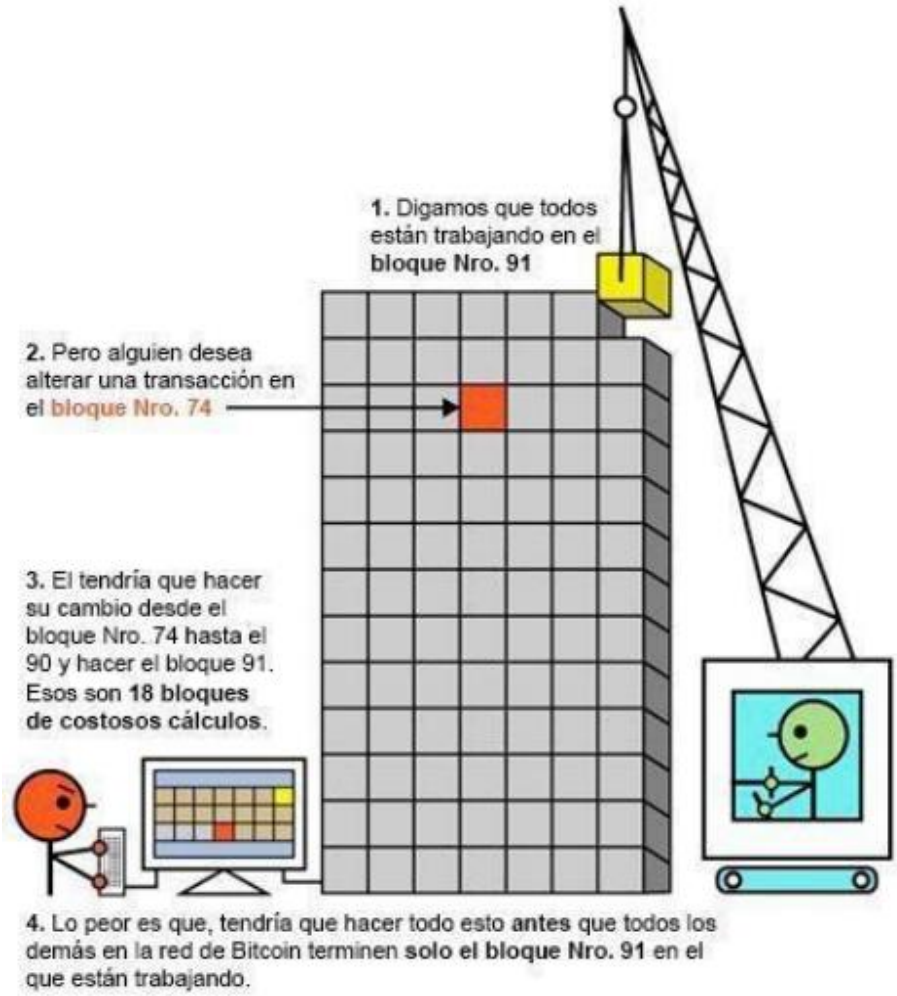
Acuerdos rápidos: Las transacciones que se realizan a través de Blockchain son rápidas por su trazabilidad, ya que puede seguir el proceso de la transacción del usuario en la red [22].
En la Figura 2 se muestra la dificultad de hackear una base de datos con Blockchain.

2.2.1.2 Tipos de Blockchain
Los desarrollos de blockchain se clasifican en los siguientes:

Blockchain Públicas: Este tipo de blockchain tiene una característica en particular, sus datos y software están habilitados al público sin importar si son usuarios, mineros o administrador. Con esta propiedad de código libre los diferentes participantes del Blockchain público pueden verificar, examinar o mejorar si lo cree pertinente, lo único

que tienen que hacer es descargar la aplicación y conectarse vía internet, por medio de un nodo actualizado, lo cual le dejará acceder a la cadena al igual que los demás miembros, también podrá replicar o minar, certificar transacciones y sugerir alguna edición o mejora. Este tipo de Blockchain tiene un buen estándar de seguridad, que se basa en un grupo de reglas que ayudan a tener un consenso sobre los cambios o mejoras. Por otro lado, la correlación de los miembros se realiza de forma anónima, lo que ayuda al software que sea más difícil que un agente malicioso pueda dañar el funcionamiento del sistema [21]. En la Figura 2 se muestra la seguridad del Blockchain.

Figura 2. Porque no se puede hackear Blockchain



Fuente: Modelo de negocio para un sistema de seguridad, respaldo y verificación digital basado en Blockchain para la gestión documental de notarias en lima moderna [21].

Blockchain Privadas: Este tipo de Blockchain tienen una central que es responsable de mantener la cadena segura, esta le da permiso para realizar las transacciones o aprobar bloques a los usuarios y administradores dentro de la red. En este Blockchain

la información y las transacciones son privadas, tampoco hay descentralización. Estas características lo hacen diferente al Blockchain público [21].

Blockchain Federadas: Este tipo de Blockchain intenta tener registros descentralizados y lograr una buena confianza en círculos laborales donde haya diferentes intereses, la información interna de este blockchain es administrada por varias entidades que son responsables de la red y la sincronización del Blockchain, esto quiere decir que la participación en la red es privada. Para acceder a este Blockchain se hace por medio de una interfaz web, el cual tiene la configuración de quienes son los usuarios y los administradores, estos miembros del Blockchain pueden revisar cada bloque de la cadena. Este tipo de Blockchain se usa normalmente en entidades gubernamentales [21].

2.2.1.3 Herramientas software con tecnología Blockchain

Geth : Esta es una herramienta basada en el lenguaje de programación Go que funciona implementando un nodo de Ethereum, esta instrumento permite desarrollar Blockchain en los sistemas operativos Linux, Windows y Mac, una de sus características destacadas es que se pueden usar 3 tipos de interfaces distintas que son el servidor JSON-RPC, la consola interactiva y la línea de comandos, este utensilio tiene las funcionalidades de crear contratos inteligentes, transferencias de tokens, comprobar el historial de bloques y extracción de tokens Ether. Todo lo anterior hace destacar a Geth como una de las herramientas más usadas para Blockchain, la cual brinda 2 formas de conectar tu sistema, una de ellas es conectarse a la red principal de Ethereum o desarrollar su propia red Blockchain [25].

Solc: Con esta herramienta se puede compilar e implementar software creados en el lenguaje Solidity sin necesidad de estar conectado a internet, ya que el paquete Solc es un módulo independiente. Solidity es un lenguaje de programación que soporta paradigma de OOP y soporta bibliotecas, por esta razón sirve para contratos inteligentes. Solc y Solidity se usan normalmente para financiamiento colectivo, votaciones, billeteras con varias firmas y subastas [25].

Remix IDE: Esta herramienta de Blockchain está enfocada en el navegador el cual respalda los procesos para el desarrollo e implementación de contratos inteligentes. Este IDE está basado en JavaScript, lo que permite el acceso a través de cualquier navegador moderno, también se puede usar alojado en el computador y una de sus características más notables es que puede crear, probar, depurar e implementar contratos inteligentes basados en lenguaje Solidity, también contiene documentación para obtener una conexión perfecta con Ethereum usando Metamask [9]. No es una herramienta para expertos, también puede usarse para aprender ya que en la web hay varios tutoriales de este IDE, el objetivo principal de este software es ser accesible para la comunidad a temas como Blockchain, Ethereum y contratos inteligentes [11].

Mist : Este software sirve para crear contratos inteligentes y también billeteras de nodo completo, es compatible con sistemas operativos que funcionan en 32 y 64 bits, pueden ser Linux, Mac o Windows, la característica particular de Mist es que usa las etiquetas de Ethereum, y una limitante de este programa es que los usuarios deben tener un lugar designado para el almacenamiento de tokens Ether y ejecución de contratos inteligentes, también deben descargar el Blockchain Ethereum completo que requiere de 1 TB, además los usuarios tienen que memorizar su contraseña de usuario porque no la pueden cambiar [25].

Truffle : Esta herramienta se usa para crear contratos inteligentes y vincularlos entre sí, también sirve para resolver problemas en el campo de Blockchain, usando Ethereum, Truffle sirve para establecer un entorno de desarrollo y poder crear software con tecnología Blockchain [25]. Este entorno de desarrollo usando la máquina virtual Ethereum tiene las siguientes características:
Un desarrollo ágil con pruebas de contrato automatizadas.
Posibilidad de migrar código y desplegar extensiones programables.
Es un IDE que sirve para compilar, vincular e implementar contratos inteligentes.
Puede crear, editar e implementar en cualquier red públicas o privadas.
Usando el estándar ERC190 puede controlar paquetes EthPM y NPM.
Sirve para ejecutar scripts externos que pueden usar scripts dentro de un entorno Truffle. [33]

Metamask : Esta herramienta se utiliza para crear una conexión entre un navegador y una cadena Blockchain Ethereum, pero tiene más funciones, también sirve como plataforma de software para usar Ether u otros ERC-20, también tiene la posibilidad de interactuar con varias redes de prueba Ethereum, al mismo tiempo sirve para usar y editar aplicaciones descentralizadas de Ethereum, los usuarios pueden combinar Coinbase y Shapeshift con Metamask para almacenar claves de tokens ETH y ERC20, luego de instalar la aplicación en el navegador, los usuarios tendrá una billetera Ethereum a su disposición [25].

Parity : Es una herramienta muy usada por desarrolladores de Blockchain facilita una infraestructura que tiene un servicio confiable y rápido, utilizando el lenguaje de programación Rust. Con esta herramienta se puede crear una red Blockchain para principiantes con uso privado o para empresas con características que se pueden personalizar [9]. Parity tiene la característica de construir las bases de Web 3.0 teniendo mucho potencial para ser escalable [26].

Blockchain Testnet: Esta aplicación sirve para probar software descentralizados antes de implementarlos. Hay 3 tipos de pruebas que se pueden realizar con este software, estos tipos de pruebas son: prueba privada, pública y CLI de ganache, estas diferentes pruebas tienen en común que se buscan errores en el software descentralizado, se pueden encontrar fallos sin necesidad de mucha inversión financiera. Testnet es una de las aplicaciones más usadas para probar Blockchain en este tiempo [25].

Ganache: Este software elaborar una red Blockchain Ethereum privada con características propias del desarrollador que la cree, con este programa se pueden inspeccionar estado y ejecutar comandos para evaluar el funcionamiento de la cadena. Este programa tiene varias características una de ellas es exploradora de bloques de datos, también tiene controles avanzados de minería. La mayoría de usuarios de Ganache, tienen este software para probar sus contratos inteligentes en el proceso de desarrollo [25].

Embark : Este programa es una plataforma que sirve para desarrollar software con bases de datos descentralizado, una de sus características es que se puede crear una aplicación HTML5 sin servidor que use tecnología descentralizada, otra característica es que se pueden crear contratos inteligentes en lenguaje JS, Embark apoya la creación de contratos en JS basados en pruebas, y se pueden administrar estos contratos en diferentes Blockchains, estos contratos pueden ser editados, migrar y se puede manejar fácilmente los múltiples contratos [25].

2.2.1.4 Lenguajes usados en Blockchain

C/C++ : El lenguaje C es catalogado como lenguaje de bajo nivel por su velocidad de binarios y la posibilidad de potabilidad de binarios, C junto a C++ son muy utilizados en diferentes proyectos por su integración entre bajo nivel y alto nivel, en Blockchain muchas empresas han empezado a implementarlo por la velocidad que brinda estos lenguajes, también la escalabilidad es un factor importante de este lenguaje, igualmente influye mucho sus bibliotecas y recursos muy modernos, pero la característica más importante es que permite hacer software multiplataforma que se puede usar en Windows, Mac, Unix y GNU/Linux. En relación a Blockchain, sus códigos se pueden implementar hasta en una RasperPi y algo para resaltar es que la primera implementación del Blockchain llamado Bitcoin está basado en C/C++ [17][3].

Solidity : Este es un lenguaje nuevo que estaba basado en los lenguajes de programación Python, JavaScript y C++, fue creado por el equipo central de Ethereum con la misión de relacionarse con la Máquina Virtual Ethereum (EVM). Este es el lenguaje más usado en 2020 para crear software con tecnología Blockchain, las características de Solidity es que soporta varias bibliotecas, tiene tipificación estática, se pueden crear paradigmas de OOP y soporta herencia. Este lenguaje de programación permite a los desarrolladores de Blockchain crear contratos inteligentes scripting inteligente, subastas ciegas, votaciones, billeteras con múltiples firmas y financiamiento colectivo, todo lo anterior se logra por sus registros inmutable y autorización de transacción [17] [3] [25].

Java : Esté lenguaje de programación tiene las características de que es muy versátil, es interpretado de manera multiplataforma y tiene alta portabilidad porque los

programas creados en Java se puede ejecutar en cualquier dispositivo computacional, no dependen de una arquitectura específica del sistema, sino que utilizan la JVM (Java Virtual Machine), este lenguaje de programación ha sido implementado para el front-end de muchos Blockchain porque permite desarrollar de forma sencilla y rápida interfaces de usuario multiplataforma, por todo lo anterior Java es uno de los mejores lenguajes para programar Blockchain [17] [3].

Python: Es un lenguaje con cadena de bloques, aunque el Blockchain que se construyen con Python tienen una tendencia a funcionar lento con operaciones criptográficas complejas por su naturaleza interpretativa, Python tiene la capacidad de crear prototipos de sus ideas rápida y fácilmente con su programación orientada a objetos [3]. Este lenguaje de programación se está convirtiendo en el más manejado para crear software de análisis de datos, estas habilidades tienen mucha demanda en el área del Blockchain, las características de manejar gran cantidad de hash hasta y controlar el volumen de transacciones, hacen de Python una herramienta muy competitiva para ser usada por empresas que empiezan a implementar el Blockchain [17].

JavaScript: Este lenguaje es usado normalmente para crear aplicaciones web, pero también ha sido usado para desarrollar Blockchain, teniendo en cuenta que JavaScript presenta problemas con escalabilidad, responsive y uso de recursos, en el ámbito de Blockchain, lo anterior lo ha limitado más que todo a desarrollar web que interactúen con software Blockchain creado en otros lenguajes más adecuados como Python, Java o C/C++ [7]. Con la innovación de NodeJs el lenguaje JavaScript se ha usado para crear cadena de bloques y los desarrolladores a veces usan este lenguaje para evitar problemas de integración con las páginas y aplicaciones web creadas con JavaScript [3].

2.2.2 Facturación electrónica

Las facturas electrónicas es un documento digital que tiene el mismo efecto legal e información que una factura en papel, pero se expide, se recibe, se rechaza y se conserva de forma electrónica. El inicio de la facturación electrónica está dividido en 2, uno es la historia de los escritos digitales y el otro es la historia del documento que comprueba el intercambio de un bien o servicio. La historia de los escritos digitales

se remonta a el inicio de internet cuando ARPANET (Advanced Research Projects Agency Network) una red de computadoras del gobierno de Estados Unidos empieza a interconectar estas computadoras con el fin de que fuera resistente a ataques tecnológicos y a la destrucción de alguno de los nodos que la componían, gracias a esta conexión y sus avances se puedo desarrollar la Word Wide Web o WWW en 1989 [1]. Luego de la creación de las conexiones entre computadoras se presencia el primer diseño de una factura electrónica realizado en 1997 por el Organismo European Articule Numbering Association (EAN-UCC), que en la actualidad lleva el nombre de Global System One (GS1). Gracias al avance tecnológico de la homogeneización en la contabilidad, que realizó la Unión Europea, se eligió a España como líder de proyecto de facturas electrónicas y se avanzó en la digitalización certificada de la hacienda fiscal [1].

2.2.2.1 Facturación electrónica en Colombia

En Colombia existe la ley 223 de 1995 que en su artículo 37 se explica textualmente "Dentro de los seis meses siguientes a la vigencia de esta Ley el Gobierno Nacional reglamentará la utilización de la factura electrónica", esta reglamentación duró más de 6 meses pero en esta ley se explicó cómo se emitirá este tipo de facturas, los parámetros de recepción, aceptación y conservación. En el año que salió esta ley ninguna empresa colombiana usaba facturación electrónica, pero en el 2013 la DIAN empezó a tomar como referencia países que ya habían avanzado en este tema como México, Brasil y Chile, y empieza a corregir los errores que encontró en la ley de 1995 buscando la obligatoriedad entre los contribuyentes para que se masifique su uso. Finalmente, en 2015 la DIAN génerо el decreto 2242 donde se vuelve obligatoria la implementación de facturas electrónicas [1].

Las características de la facturación electrónica en Colombia son:

- Utiliza el formato electrónico de generación XML estándar establecido por la DIAN.
- Lleva la numeración consecutiva autorizada por la DIAN.
- Cumple los requisitos del 617 ET y discrimina el impuesto al consumo cuando es el caso.

- Incluye la firma digital o electrónica para garantizar autenticidad e integridad y no repudio de la factura electrónica, de acuerdo con la política de firma adoptada por la DIAN.
- Incluye el Código Único de Factura Electrónica CUFE [1].

En la Figura 3 se muestra cómo está estructurada una factura electrónica colombiana.

Figura 3. Ejemplo de factura electrónica colombiana

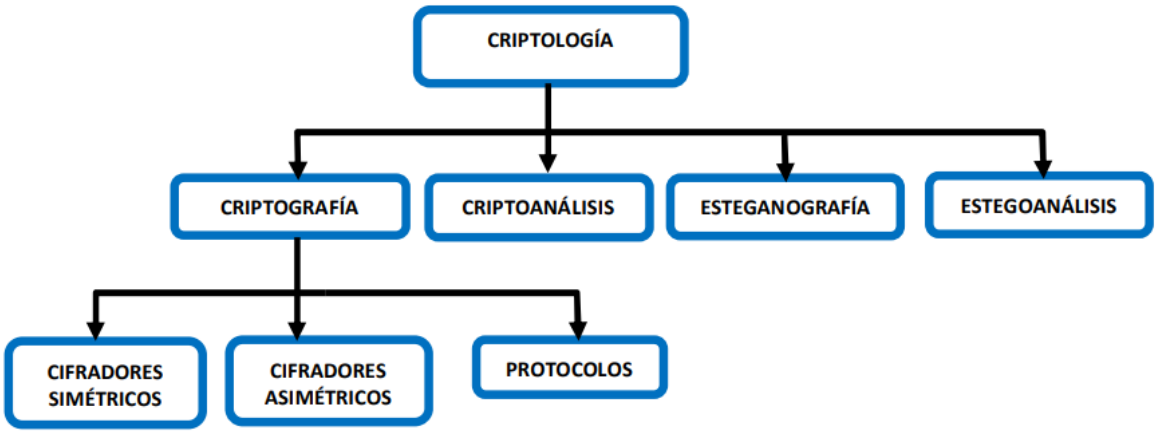
Fuente: Implementación Facturación electrónica en Colombia [1]

2.2.3 CRIPTOGRAFÍA

El término criptografía proviene del giego kryptós, “escondido”, y graphos, “escritura”, es una rama de la criptología, y su función es encriptar mensajes que están compuestos de signos comunes a unos datos que aparentemente no contenga información en ellos, en otras palabras, es convertir un texto a un escrito que sea indescifrable para quien no posee la clave de descifrado. El método de cifrado se llama encriptación y el método de descifrar se llama desencriptación. La criptografía

es un arte muy antiguo y se puede comprobar en los jeroglíficos secretos usados en el año 2000 AC, y se continúa usando la criptografía en las comunicaciones de la actualidad [23]. La criptografía intenta mantener segura la información contando con 4 pilares que son: confidencialidad, integridad, autenticidad y no repudio. Además,la criptografía se divide en encriptación de flujo que encripta señales de los móviles y de wifi y la otra es encriptación por bloques que se encarga de cifrar sesiones y firmas digitales [18]. En la Figura 4 se muestra el árbol de antepasados de la criptografía.

Figura 4. La criptografía como parte de la criptología



Fuente: Algoritmos de encriptación de clave asimétrica [23].

2.2.3.1 Tipos de Criptografía

Existen 3 tipos de Criptografía que son:

Algoritmos Simétricos (o de clave privada): Este tipo de encriptación consiste en que el mensaje pasa por una clave privada de cifrado antes de ser enviada al destinatario, luego de que se recibe el mensaje se procede a usar la misma clave privada para desencriptar la información para poder entender el mensaje. Tanto el remitente como el destinatario se ponen de acuerdo para saber cuál es la clave privada. Hasta 1976 todas las encriptaciones eran simétricas y en la actualidad se usa para el cifrado de datos y verificación de integridad del mensaje [23]. Una característica positiva que tiene este tipo de criptografía es que son muy rápidos al momento de encriptar y desencriptar la información, normalmente se usan claves que tienen desde 128 hasta 256 bits con esto se adquiere categorías de cifrado de alta

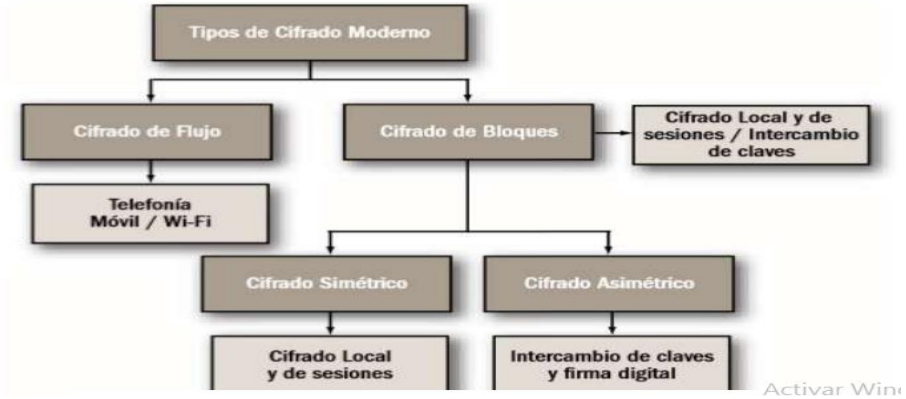
calidad. Una característica negativa de este tipo de criptografía es que son débiles ante ataques de fuerza bruta, ya que luego de interactuar con esta encriptación se puede lograr con la clave privada para poder desencriptar la información sin ser el emisor o el destinatario de estos datos [18].

Algoritmos asimétricos (o de clave pública): Este tipo de criptografía usa 2 claves por cada usuario, una para encriptar que se llama clave pública y otra para desencriptar que es la clave privada. El inicio de la criptografía asimétrica empezó buscando una forma de intercambiar las claves simétricas. Los investigadores Diffie y Hellman propusieron la teoría para cifrar y descifrar de manera asimétrica en el año 1976 pero fue hasta 1978 que Rivest, Shamir y Adleman propusieron el método RSA, el cual se basa en la imposibilidad computacional de factorizar números enteros grandes. Este tipo de criptografía se usan normalmente para crear firmas digitales, establecer claves y también en cifrados clásicos [23]. Un punto negativo de las claves asimétricas es que dependen de complicados cálculos matemáticos [18].

Esquemas híbridos En criptografía: Este tipo de criptografía tiene la cualidad de que funciona como un sistema de cifrado de clave pública con la velocidad de un sistema de clave privada, y en casos de mensajes extremadamente largos, la mayor parte del trabajo de encriptación y desencriptación de datos los hace la clave privada. El cifrado de clave pública se usa solo para cifrar y descifrar un valor de clave corte, esto ayuda a que este tipo de criptografía tenga las características positivas de los algoritmos de clave privada y los de clave pública [23].

En la Figura 5 se muestra los tipos de cifrados modernos y su uso.

Figura 5. Tipos de cifrado moderno



Fuente: Técnicas de encriptación para mejorar la seguridad en la transferencia de archivos en un entorno fiable [18].

2.2.3.2 Encriptaciones más usadas

Hay varios algoritmos de encriptación tanto de clave privada como de clave pública, algunos de los primeros algoritmos de clave privada más destacados son: DES, triple DES, AES, IDEA y Blowfish. Los algoritmos que más se han destacado de clave pública son RSA y el Gamal [18]. En la actualidad también hay algoritmos híbridos que se utilizan con frecuencia. A continuación, se muestra en la Figura 6 el funcionamiento de algunos de estos algoritmos de encriptación.

Figura 6. Algoritmos de cifrado simétrico más conocidos

Algoritmo	Tamaño de Claves (Bits)	Tamaño de Bloque (Bits)	Número de Etapas	Aplicaciones
DES	56	64	16	SET Kerberos
3DES	112 o 168	64	48	PGP, S/MIME
AES	128, 192 o 256	128	10, 12 o 14	
IDEA	128	64	8	PGP
Blowfish	Variable hasta 448	64	16	Varias
RC5	Variable hasta 2048	64	Variable hasta 256	Varias

Fuente : Técnicas de encriptación para mejorar la seguridad en la transferencia de archivos en un entorno fiable [18].

Data Encryption Standard (DES): Es un algoritmo de encriptación de clave o también llamado encriptación asimétrica, funciona dividiendo la información en bloques de datos con tamaño igual a 56 bits, y luego pasando estos bloques por medio de funciones matemáticas que están compuestas por sustituciones, lineales y no lineales. Este algoritmo fue creado en 1997 y en 1998 fue atacado, durante 56 horas

en un primer momento y luego 22 horas, creando la necesidad de un algoritmo más robusto. En 2001 este algoritmo es sustituido por AES [24].

Triple DES: Fue creado en 1998 por la empresa IBM y consiste en 3 funciones DES encapsuladas en 1 solo software, teniendo en cuenta que DES estaba compuesto por una clave de 56 bits. El triple DES está compuesto por una clave de cifrado de 168 bits, este algoritmo de encriptación fue reemplazado por el algoritmo AES que puede ser 6 veces más rápido que triple DES [18].

International Data Encryption Algorithm (IDEA): Este algoritmo de encriptación fue creado en 1991 por James Massey de ETH Zurich y Xuejia Lai con el objetivo de reemplazar el algoritmo DES. IDEA funciona cifrando bloques de 64 bits, con claves de 128 bits y un total de 8 rondas. Este algoritmo fué utilizado en las primeras versiones de PGP, que fué uno de los softwares de cifrado más comunes usado a finales del año 1991. IDEA funciona ejecutando operaciones algebraicas no conmutativas de grupos algebraicos diferentes como son XOR, sumas y multiplicaciones, todas estas realizadas sobre bloques y las subclaves usadas [18].

DSA – Digital Signature Algorithm: Empezó siendo un simple algoritmo de encriptación en asimétrica en 1991 y luego se convirtió en un estándar del Gobierno Federal de los Estados Unidos de América o FIPS para firmas digitales, funciona trabajando con 2 grupos cíclicos uno de 1024 bits de longitud y el otro más pequeño de 160 bit, de esta forma se obtienen firma de longitud más corta. Una característica negativa de este software es que requiere de mucho procesamiento computacional para cifrar grandes cantidades de información [23].

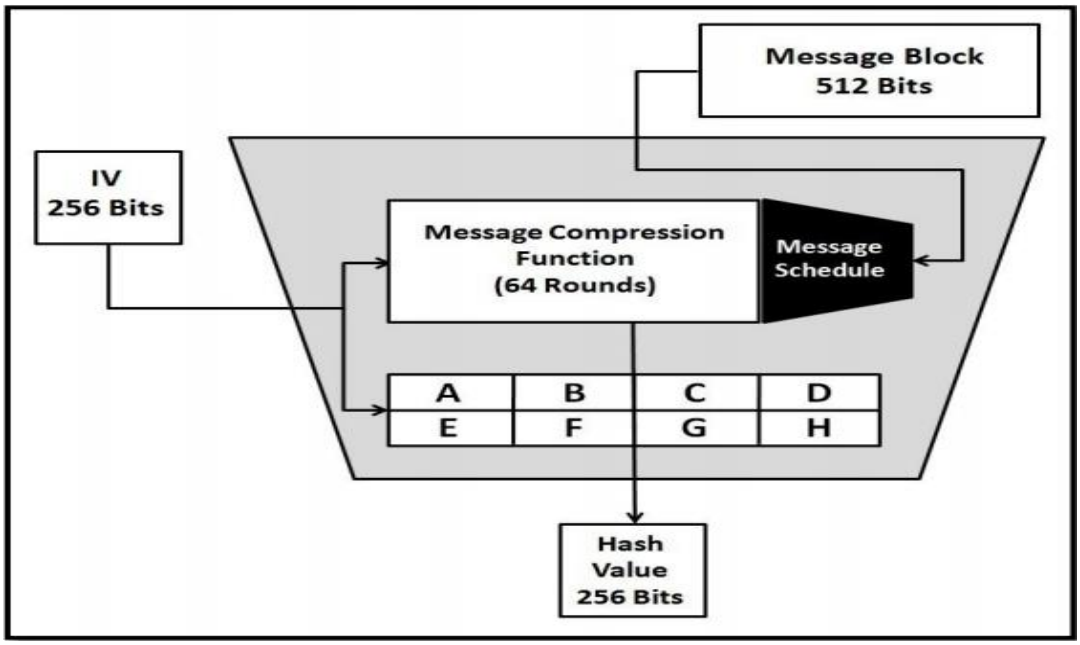
Blowfish: Este algoritmo de encriptación fue creado por Bruce Schneier en 1993, cifrando bloques que operan sobre 64 bits con unas claves que pueden ser de 32 a 448 bits, este tipo de encriptación funciona ejecutando 16 rondas y aplicando el cifrado Feistel en la que se ejecuta una clave dependiendo de las Cajas S, que es un componente básico de clave simétrica. El objetivo de la creación de este algoritmo fue reemplazar a DES y a IDEA, pero Blowfish no llegó a convertirse en estándar, en la actualidad el código de Blowfish está pública a disposición de los usuarios [18].

RSA – Rivest, Shamir y Adleman: Este algoritmo se creó por Rivest, Shamir y Adleman en el año 1978 por las siglas de sus creadores se le pone el nombre RSA, este es el primer algoritmo efectivo de clave pública o asimétrico, funciona aprovechando el problema de factorización de números enteros, usando la operación inversa de la factorización, para entender su funcionamiento es prudente conocer el teorema de euler y la función phi de euler ya que en este algoritmo son muy importantes. Normalmente se usa para crear cifrados de pequeñas cantidades de datos como las claves o firmas digitales. Este algoritmo no fue creado con el fin de reemplazar los algoritmos simétricos o de clave privada, porque es de procesamiento lento con grandes cantidades de información [23]. Los algoritmos creados antes de RSA fueron vulnerados por ataques maliciosos excepto El Gamal y RSA por la composición de sus algoritmos que se pueden usar para cifrados o firmas electrónicas tanto de encriptación como de desencriptación usando su clave privada [18].

2.2.3.3 SHA256

Este algoritmo de encriptación llamado SHA256 fue creado por la NSA (Agencia de Seguridad Nacional de los Estados Unidos) y funciona adquiriendo una entrada con una longitud de menos de 264 bits, y luego de tener esta entrada lo convierte en un bloque del tamaño de 512 bits que se representa como una secuencia de 16 palabras de 32 bits. El bloque de 512 bits ingresa a una función de compresión de mensaje en palabras de 32 bits por medio de un programador de mensajes, este programador expande el bloque de mensajes de 512 bit en 64 palabras de 32 bits. Se hacen 64 rondas del procedimiento anteriormente explicado para comprimir el mensaje inicial de 265 bit a uno de 32bits. Este algoritmo de hash SHA256 se puede comparar a una encriptación de bloque con un tamaño de mensaje de 256 bits y una clave de 512 bits. Este algoritmo se hizo famoso cuando Bitcoin lo utilizo como la encriptación base para sus bloques de datos Blockchain porque este tipo de cifrado es muy difícil de romper por computadoras y ataques informáticos de la actualidad [28]. En la figura 7 se muestra cómo funciona el algoritmo SHA256.

Figura 7. Descripción del algoritmo SHA256

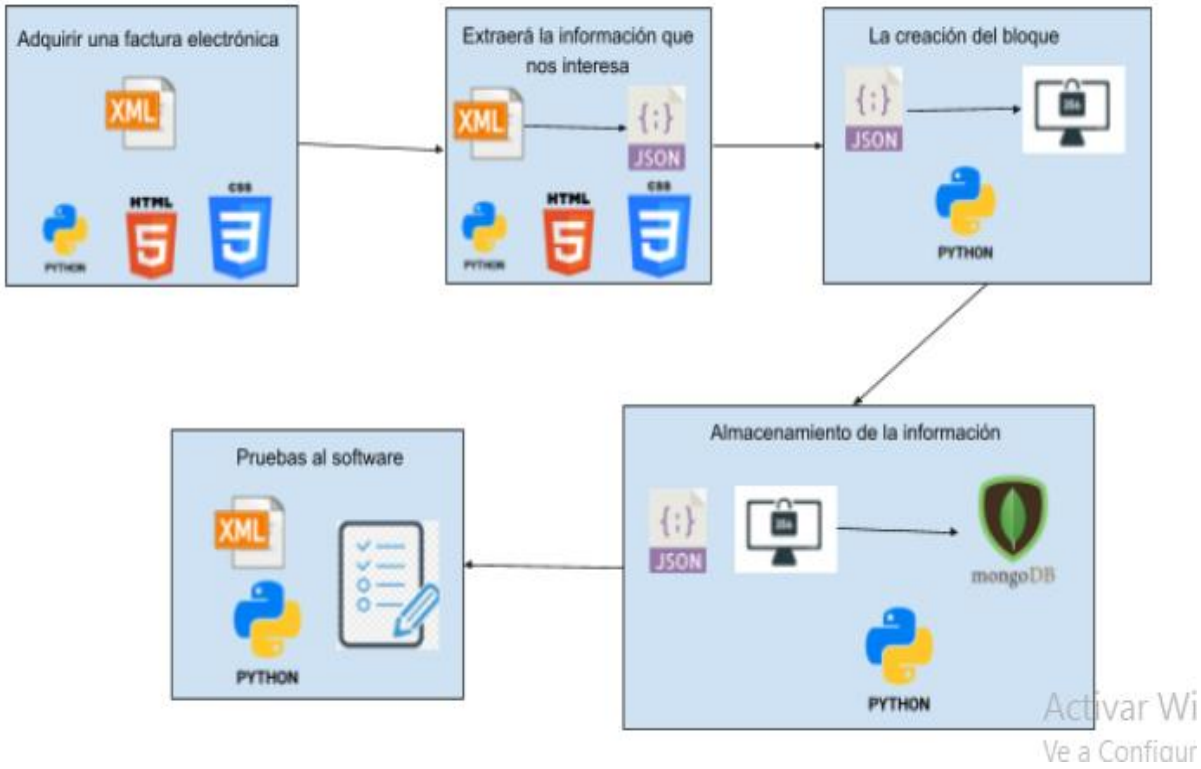


Fuente: Optimising the SHA256 hashing algorithm for faster and more efficient Bitcoin mining [28]

3. METODOLOGÍA

En este capítulo del documento se muestra la metodología para diseñar e implementar una herramienta web con un sistema de Blockchain para la facturación electrónica colombiana. Esta empieza con el subproceso de adquirir una factura electrónica en formato XML, luego se le extraerá la información que nos interesa para posteriormente encriptarla, después de tener la información de interés se continúa con la creación del bloque para agregarlo a la cadena ya existente o en el otro caso que este sea el bloque inicial. La metodología explicada anteriormente se representa en el diagrama de la Figura 8 donde se muestra las etapas de cada proceso.

Figura 8. Diagramas de bloque de la metodología para un sistema de facturación electrónica usando tecnología Blockchain.

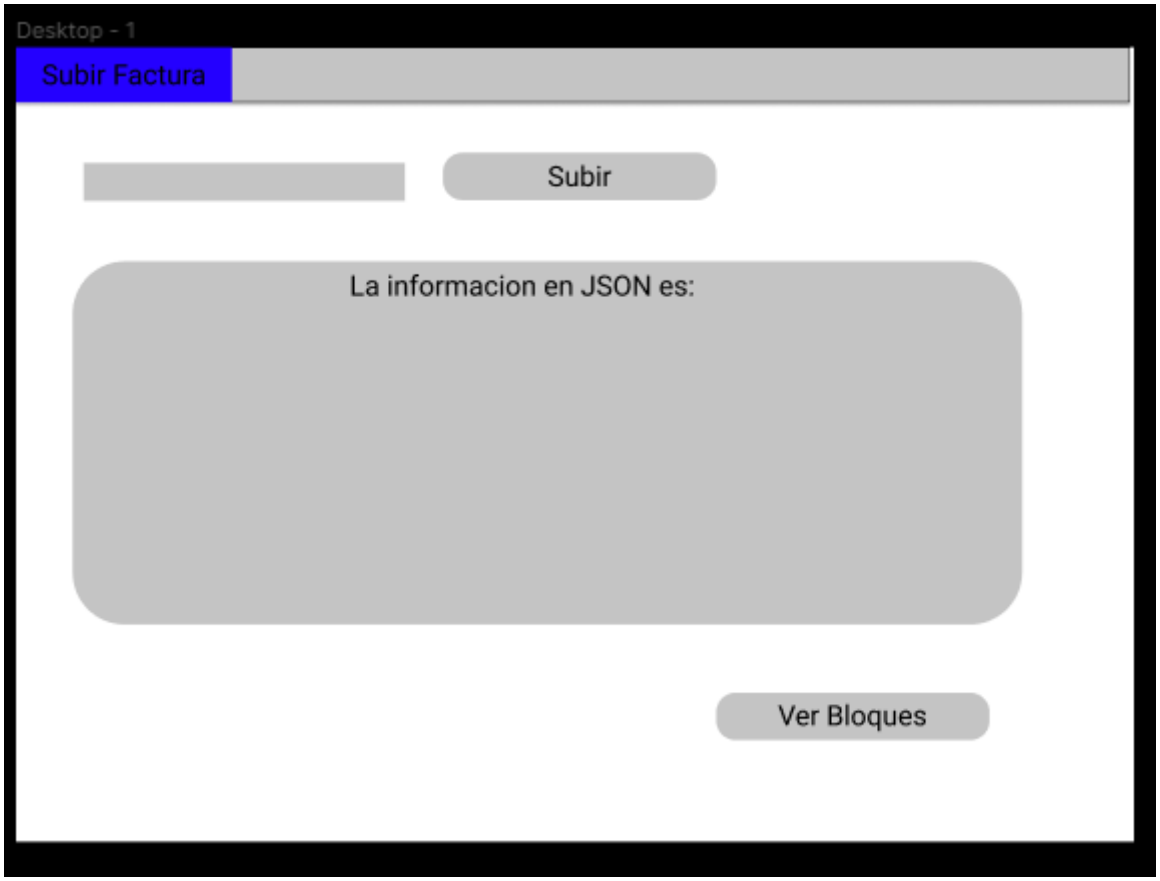


3.1 DISEÑO DE LAS INTERFACES DE USUARIO

Las interfaces son amigables con el usuario y fáciles de utilizar, la primer interfaz que tendrá el programa es la de subir el archivo, que en una barra superior mostrará 2 botones que nos ayudarán a navegar por el software, el primero sirve para entrar a la interfaz de subir archivo y el segundo para visualizar los bloques ya creados, dentro de la interfaz de subir archivo también contendrá los botones de cargar archivo que ayudará a subir las facturas electrónicas XML y los convierte en JSON, también creará

su respectivo hash encriptado. El botón de cargar archivo mostrará un mensaje cuando el XML se haya cargado correctamente, y cuando el archivo se haya convertido a JSON se mostrará la información en el centro de la página web. Por último en la parte inferior de la página está el botón de Ver bloque el cual nos muestra los bloques creados anteriormente, entonces este botón ejecuta un algoritmo de conexión con la base de datos y subirá la información que anteriormente se creó en la base de datos MongoDB. El diseño de la interfaz se muestra en la Figura 9.

Figura 9. Esquema de interfaz de la sección subir archivo



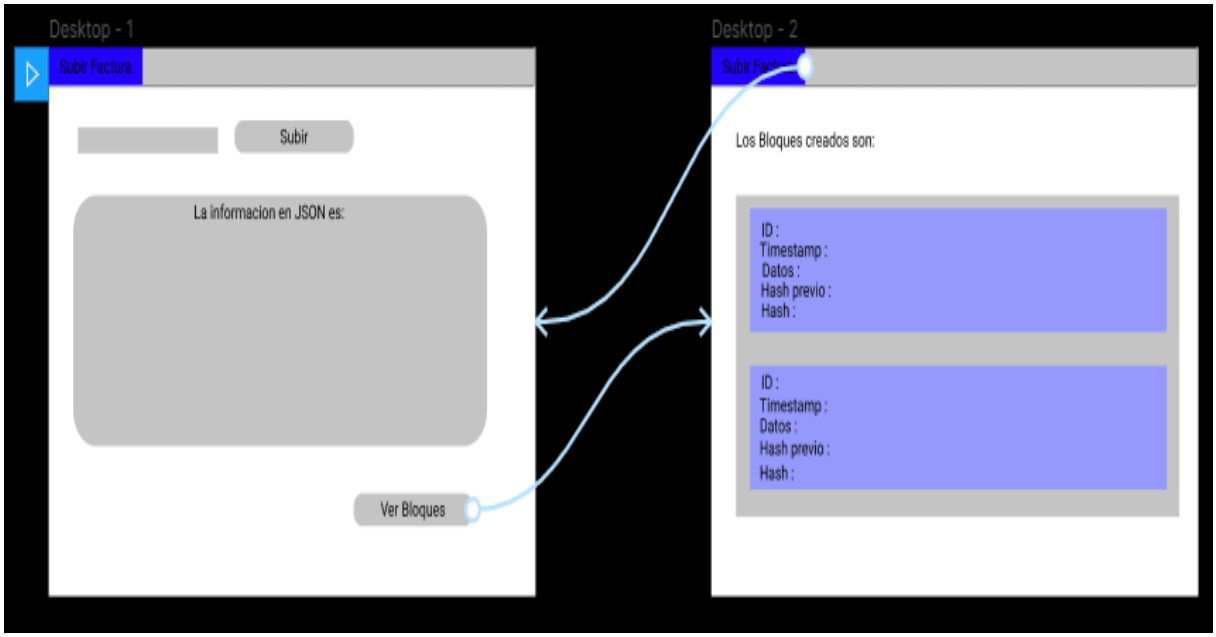
La segunda interfaz que tendrá el programa es la de Ver bloques, el botón que nos muestra los bloques creados está en la parte inferior de la página “Subir Archivo” el cual sirve para visualizar los bloques ya creados, dentro de la interfaz de ver bloques se mostrarán los bloques de datos creados con anterioridad los que llevan la información seleccionada de las facturas electrónicas en XML también mostrarán el hash previo de cada bloque y el hash actual de cada uno de los bloques como se muestra en la siguiente Figura 10.

Figura 10. Esquema interfaz de la sección ver bloques

Al momento de interactuar con las interfaces de usuario, estando en la interface 1 de seleccionar archivo, si se da click en el botón que sirve para subir archivo se habilita el recuadro para seleccionar la factura electrónica y si se da click en el botón “Subir” se cargará la factura electrónica y la información se almacena en variables internas y luego de dar click en el botón “Ver Bloques” se abre la interfaz 2 de la sección de ver bloques donde se podrán visualizar los Bloques creados.

Luego de estar en la interfaz 2 si se da click en el botón que está en el parte superior llamado “Subir Factura” se devolverá a la interfaz 1 que lleva el nombre del botón. Lo que se explica en este párrafo se muestra en la Figura 11.

Figura 11. Diseño de comunicación entre interfaces

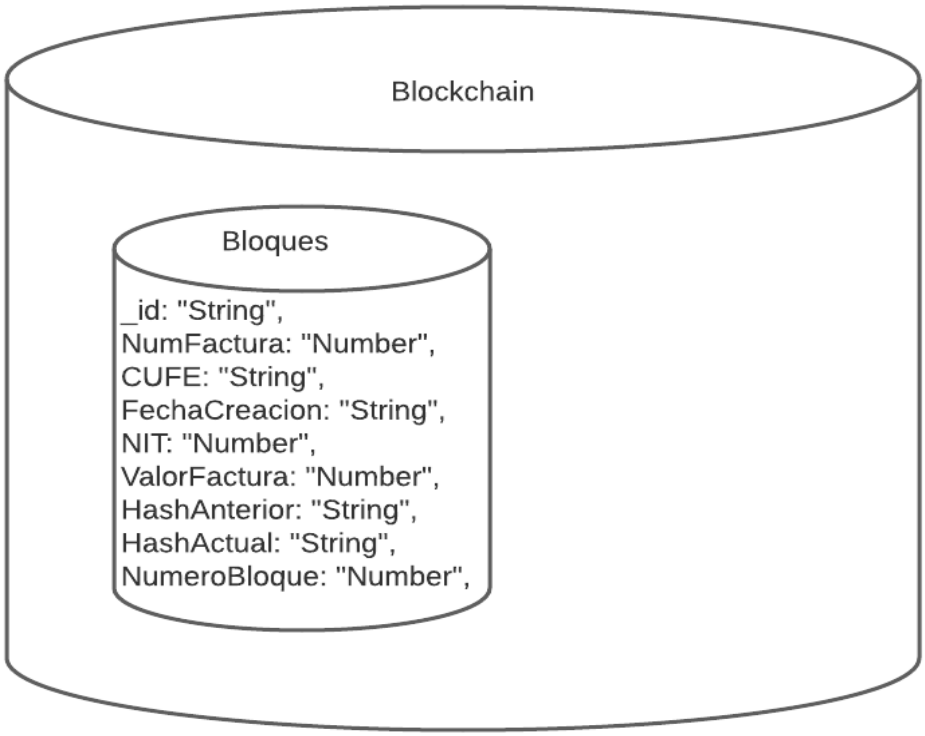


3.2 DISEÑO DE BASE DE DATOS

El motor de base de datos que se usa es MongoDB, el nombre de la base de datos que se utiliza es “Blockchain” y dentro de esta base de datos está la colección llamada “Bloques” el cual tiene una variable id que se pone automáticamente cuando se ingresan datos a la colección. Además,se tiene el número de factura, el NIT de la empresa que emite la factura electrónica y el número del bloque que se manejan en variables de tipo “Number” que hace referencia a los números, también tiene el código CUFE, la fecha de creación, el hash anterior y el hash actual que son variables de tipo string.

Se escoge este motor de base de datos por la posibilidad de manejar información descentralizada y también por la seguridad que mantienen las bases de datos NoSql, ya que evitan varios tipos de intrusión como lo son las Sql injections. Por otro lado, se escoge MongoDb por la estructura de datos en JSON con el que se pueden manejar grandes cantidades de información. En la Figura 12 se muestra el diagrama de la base de datos.

Figura 12. Diseño de la base de datos Blockchain con coleccion Bloques



3.3 LECTURA DE FACTURA ELECTRÓNICA EN XML

Las facturas electrónicas son creadas por cada empresa en particular y la estructura de las facturas electrónicas son validadas por la DIAN, la cual verifica que estos archivos cumplan con todos los requisitos que impone la ley de Colombia, algunos datos que verifica la Dian es que estas facturas están en formato XML, que tenga el CUFE entre su información, la fecha de creación, el NIT de la empresa y el valor de la factura. La adquisición de estas facturas electrónicas en el software se hará por medio del lenguaje de programación Python acompañado de HTML y CSS, el cual contará con un segmento de código donde al dar click en un botón se habilitará la posibilidad de seleccionar un archivo y luego de seleccionarlo se subirá el archivo a la raíz del Software como se muestra en la siguiente Figura 13.

Figura 13. Diagramas de bloque de la metodología para almacenar las facturas electrónicas

3.4 CONVERSIÓN DE ARCHIVO XML A JSON

Luego de tener la factura electrónica XML en la raíz del programa se continúa escogiendo la información de importancia de la factura para posteriormente organizarla en un JSON. Lo anterior funciona con una interfaz de usuario organizada en HTML y CSS con botones amigables para su uso, que tendrían una lógica en Python de extracción y conversión de información. Es necesario tener la información de la factura electrónica en formato JSON para poder usarla en la creación del Blockchain y el almacenamiento en la base de datos MongoDB. En la Figura 14 podemos ver el diagrama de bloques de la conversión de XML a JSON.

Figura 14. Diagramas de bloque de la metodología para convertir XML a JSON

3.5 ENCRIPTACIÓN DE LA INFORMACIÓN

Luego de tener la información en JSON extraída de la factura electrónica XML se procede a usar el algoritmo de encriptación SHA256 el cual coge la información que está en JSON y la convierte en un código de 256 bits representado en un hexadecimal que es único para el conjunto de información seleccionado, donde este código es

guardado en una variable que se usará más adelante. Un cambio en alguno de los datos seleccionados para ingresar al SHA256 puede generar un cambio en el código hexadecimal que tendrá la salida del software. La encriptación SHA256 es una forma de encriptar más robusta que el SHA-1 que poseen las facturas electrónicas antiguas en Colombia. El SHA256 no ha presentado vulnerabilidades en su seguridad y teniendo en cuenta que el SHA-1 ya ha presentado vulnerabilidades es mejor trabajar con el SHA256 para fortalecer la seguridad de las facturas electrónicas. La metodología de encriptación se muestra en la Figura 15.

Figura 15. Diagramas de bloque de la metodología para encriptar la información de la facturación electrónica

3.6 CONFIGURACIÓN DE BASE DE DATOS MONGODB

Luego de encriptar la información proveniente de la factura electrónica se procede a crear la conexión desde el software hacia la base de datos MongoDB usando la librería de Python llamada PyMongo. Posterior a esto, se coloca el nombre de la base de datos, para posteriormente crear la colección y anexar a la tabla la información adquirida de la factura electrónica junto al hash del bloque anterior y el hash del bloque reciente. Dado el caso de que este sea el primer bloque de información creado en el software tendrá el nombre de Genesis y no tendrá el hash previo. En la Figura 16 se muestra la metodología que se usó para configurar la base de datos.

Figura 16. Diagramas de bloque de la metodología para encriptar la información de la facturación electrónica

3.7 DISEÑO DE PRUEBAS DE SOFTWARE

En esta etapa se ejecutan pruebas para comprobar la seguridad del software mejorando el almacenamiento de la factura electrónica, se hará una prueba donde se muestra el cambio en el HashActual de un bloque en el cual se modifique 1 solo dato de toda la información que contiene. La segunda prueba muestra el cambio de cada uno de los HashActuales de cada uno de los bloques que se van creando, siendo muy diferente el hash de cada uno de los bloques que se crean con el dato cambiado, lo cual nos ayuda a entender que los cambios afectarán los hash futuros.

4. RESULTADOS

A continuación, se presentan los resultados obtenidos luego de implementar la metodología explicada en el anterior capítulo. Los resultados muestran la creación de la interfaz, la estructura de la base de datos, la adquisición y procesamiento de las facturas electrónicas, la conversión de archivos XML a JSON, la encriptación de la información, el funcionamiento de la base de datos y las pruebas realizadas. El código completo de software creado se encuentra en el link: https://github.com/guevaraStian/FacturaElectronicaBlockchain .

4.1 CREACION DE LAS INTERFACES DE USUARIO

Este software tiene 2 interfaces, la primera es donde se sube la factura electrónica y se convierte en un bloque Blockchain y la otra interface es donde se muestra los Bloques creados con la información adquirida de las facturas electrónicas. En la Figura 17 se muestra una parte del código de la interface 1.

Figura 17. Interfaz de subir factura electrónica

```
<h1>Facturacion elctronica con Blockchain</h1>
<form action="/upload" method="POST" enctype="multipart/form-data">
  <input type="file" name="archivo" class="btn btn-secondary" />
  <input type="submit" value="Subir" class="btn btn-secondary"/>
</form>
<br>


<table class="table table-responsive table-bordered">
  <thead class="bg-danger">
    <tr>
      <th scope="col" >Partes del documento</th>
      <td scope="col" >Informacion dentro del documento</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row" class="bg-secondary">NumFactura</th>
      <td scope="row" class="bg-secondary">{{NumFactura}}</td>
    </tr>
```

En la vista 2 de mostrar los bloques, podemos ver como esta creado el IF con request.method que se activa cuando POST se activa, luego se hace la conexión a la base de datos y a la colección para posteriormente hacer una consulta “find” en la colección “col” y su respuesta se guarda en la variable “listabloques”. En la Figura 18 se muestra lo explicado en el párrafo anterior.

Figura 18. Configuración del botón que direcciona a la interface 2 que muestra los Bloques creados

```
94  @app.route("/mostrar", methods=['POST'])
95  #Luego de que activa el metodo 'POST' con la ruta "/mostrar" corre la siguiente funcion
96  def subirinfo():
97      if request.method == 'POST':
98          #Se conecta a la base de datos "Blockchain" y la coleccion "Bloques"
99          client = MongoClient('localhost', port=27017, username='', password='')
100         db = client['Blockchain']
101         col = db['Bloques']
102         #Se consulta los bloques creados y se guarda en la variable "listabloques"
103         listabloques = col.find()
104
105
106     return render_template('verbloques.html', listabloques=listabloques)
```

Luego de que el usuario use el boton “Ver bloques”, se carga la vista de bloques creados que tiene en su interior un “container” y dentro de este hay un “body” que contiene un FOR donde “i” es un bloque dentro de “listabloques” y en cada uno de los bloques se extrae los datos en particular como por ejemplo “NumeroBloque” y “NumFactura”.

El FOR se cierra al final de la muestra de los datos y esto hace que se muestre cada uno de los bloques por separado, las clases que se usan en cada div son extraídas de bootstrap. En la Figura 19 se muestra lo explicado anteriormente.

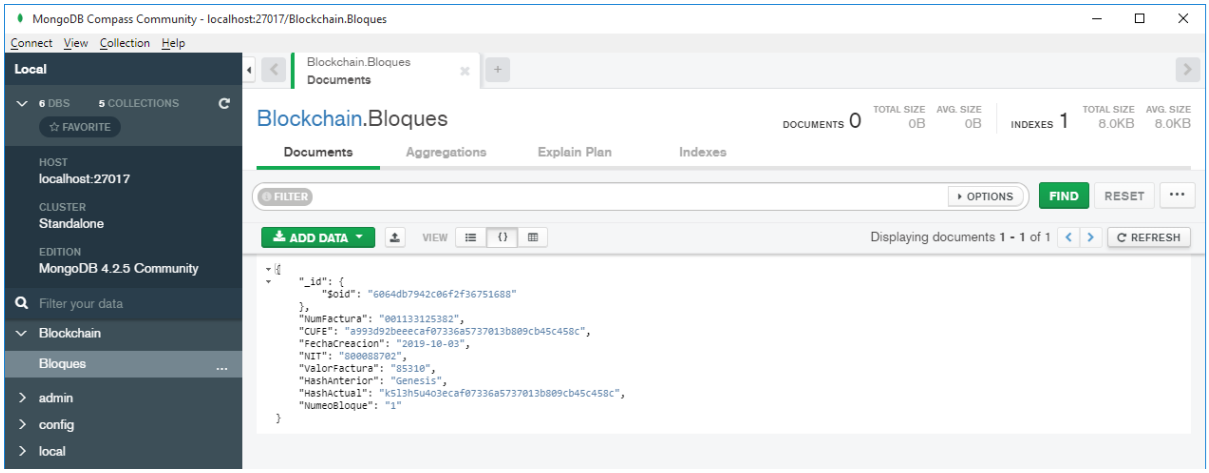
Figura 19. Interfaz de mostrar bloques creados

```
<!-- codigo para mostrar todos los bloques -->
    {% block result %}

  <div class="container" id="test">
            <tbody id="body">
                {% for i in listabloques%}
                <div class="card bg-secondary mb-3" >
                  <div class="card-body">
                <tr>
                    <td> <th> NumeroBloque = </th> {{ i["NumeroBloque"] }}</td>
                    <br>
                    <td> <th>NumFactura =</th> {{ i["NumFactura"] }}</td>
                    <br>
                    <td> <th>CUFE = </th>{{ i["CUFE"] }}</td>
                    <br>
                    <td> <th>FechaCreacion = </th> {{ i["FechaCreacion"] }}</td>
                    <br>
                    <td> <th>NIT = </th>{{ i["NIT"] }}</td>
                    <br>
                    <td> <th>ValorFactura = </th> {{ i["ValorFactura"] }}</td>
                    <br>
                    <td> <th>HashAnterior = </th> {{ i["HashAnterior"] }}</td>
                    <br>
                    <td> <th>HashActual = </th> {{ i["HashActual"] }}</td>
                </tr>
              </div>
            </div>
                {% endfor %}
    </div>

{%endblock%}
```

4.2 ESTRUCTURA DE BASE DE DATOS

El motor de la base de datos es MongoDB y la base de datos creada se llama “Blockchain”, y dentro de esta se crea la colección llamada “Bloques”, como se muestra en la Figura 20, la creación de la estructura de la base de datos usando Compass, una herramienta de Mongo DB.

Figura 20. El motor de base de datos MongoDB queda con la base de datos Blockchain y la colección Bloques



La información extraída de las facturas electrónicas queda almacenada en MongoDB como un texto JSON. MongoDB queda en espera de la creación de otro bloque de información que llevará un hash previo con toda la información del bloque anterior. En la Figura 21 se muestra la colección “Bloques” con el JSON que tiene la información extraída de la factura electrónica.

Figura 21. Dentro de la base de datos quedan ingresados los datos en formato JSON


```
{
    "_id": {
        "$oid": "6064db7942c06f2f36751688"
    },
    "NumFactura": "001133125382",
    "CUFE": "a993d92beeecaf07336a5737013b809cb45c458c",
    "FechaCreacion": "2019-10-03",
    "NIT": "800088702",
    "ValorFactura": "85310",
    "HashAnterior": "Genesis",
    "HashActual": "k5l3h5u4o3ecaf07336a5737013b809cb45c458c",
    "NumeoBloque": "1"
}
```


4.3 ADQUISICIÓN Y PROCESAMIENTO DE FACTURA ELECTRÓNICA EN XML

Se desarrolló con una estructura HTML conectada con un código en Python,donde el HTML tiene una acción llamada “/upload” que es una función dentro del código Python, también usa el método “POST” que sirve para enviar información texto, datos

binarios, imágenes, grabaciones de audio y diferentes tipos de archivos, no se usó el método “GET” porque solo sirve para enviar texto por medio de la URL. En la Figura 22 se muestra cómo está organizado “formulario.html” donde se lee la factura electrónica.

Figura 22. Etiquetas HTML de la lectura de la factura electrónica

```
<p><label>Selecciona un nuevo archivo!</label></p>
<form action="/upload" method="POST" enctype="multipart/form-data">
    <input type="file" name="archivo">
    <input type="submit">
</form>
```

Se importaron las librerías requeridas por Python, que se usarán en la subida de la factura electrónica. En la Figura 23 se muestran las librerías que se usan en el código Python para subir archivos.

Figura 23. Librerías Python importadas para subir archivos

```
# Importamos todo lo necesario
import os
from flask import Flask, render_template, request
from werkzeug.utils  import secure_filename
```

Para iniciar el programa se crea el objeto Flask, luego se le configura una variable “UPLOAD_FOLDER” con la dirección de una carpeta llamada “FacturasElectronicas. De la línea de código 17 a la 19 se inicializa la aplicación. En la línea 21 del código se declara una ruta URL base “/” y se crea una función que retorna la página con el nombre “formulario.html”. En la línea 26 del código se declara una ruta llamada “/upload” con el método POST y cuando el método pedido por la página “formulario.html” es POST se corre la función donde se obtiene el input del archivo y se guarda en una variable “f”. Luego se usa la función secure_filename de la librería “werkzeug.util” para guardar la factura electrónica en una variable filename y por último se guarda el archivo en la variable “UPLOAD_FOLDER” que hace referencia a la dirección dentro del software llamada “FacturasElectronicas”. En la Figura 24 se muestra lo explicado en el párrafo anterior.

Figura 24. Implementación del código Python para subir la factura electrónica XML

```
12  # instancia del objeto Flask
13  app = Flask(__name__)
14  # Carpeta de subir archivos
15  app.config['UPLOAD_FOLDER'] = './FacturasElectronicas'
16
17  if __name__ == '__main__':
18  # Iniciamos la aplicación
19      app.run(debug=True)
20
21  @app.route("/")
22  def upload_file():
23      # si el link termina en / se muestra la plantilla "formulario.html"
24      return render_template('formulario.html')
25
26  @app.route("/upload", methods=['POST'])
27  def uploader():
28      if request.method == 'POST':
29          # obtenemos el archivo del input "archivo"
30          f = request.files['archivo']
31          filename = secure_filename(f.filename)
32          # Guardamos el archivo en el directorio "FacturasElectronicas"
33          f.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
34          # Retornamos una respuesta satisfactoria
```

4.4 CONVERSIÓN DE ARCHIVO XML A JSON

Luego de tener el archivo XML de la factura electrónica guardada en 1 variable llamada “filename” se procede a importar la librería LXML de Python para poder convertir la variable con el archivo XML en un árbol XML con cadena de datos que luego se usará para extraer información de este árbol XML. En la Figura 25 se muestra la librería que se importa y como se hace la importación.

Figura 25. Importar la librería LXML de Python

```
7   from lxml import etree
```

Después de importar la librería y de tener el archivo XML en una variable se extrae el árbol XML y se almacena en una variable llamada “doc” como se muestra en la línea 41 de la Figura 26. Se continua con la extracción de algunos textos que se almacenan dentro de las etiquetas que están dentro del árbol XML. En la Figura 26 se muestra cómo se guarda esa información específica en las diferentes variables para que se

pueda almacenar en una base de datos MongoDB. Se usa la opción findtext para extraer el texto dentro de la etiqueta que está guardada en el árbol XML llamado “doc”.

Figura 26. Guardar la información dentro de las etiquetas en su respectiva variable

```
f = request.files['archivo']
#Se guarda en la variable doc la inforacion que esta en el archivo recien subido
doc = etree.parse(f)
filename = secure_filename(f.filename)
f.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))

#Se guarda en variables la informacion que esta en cada etiqueta XML
NumFactura = doc.findtext("fac/numfactura")
CUFE = doc.findtext("fac/CUFE")
FechaCreacion = doc.findtext("fac/startdate")
NIT = doc.findtext("fac/nitempresa")
ValorFactura = doc.findtext("fac/totalvalor")
```

Luego de hacer la conexión a la base de datos y de seleccionar la colección “Bloques” en la variable “col” se procede en la línea 47 a evaluar si la colección está vacía y si es así se guarda en la variable “colvacia” un True, de lo contrario un False. En la línea 50 se declara el IF para cuando la colección está vacía, dentro de este IF el HashAnterior será igual a “Genesis” y el NumeroBloque será igual a 1, posteriormente se convierte el NumeroBloque en String para subirlo a la base de datos. Lo explicado anteriormente se muestra en la Figura 27.

Figura 27. Verifica si el documento que se va a guardar es el 1ro que estará dentro de la colección “Bloques”

```
41      #Conexion a la base de datos "Blockchain" y seleccion de la coleccion "Bloques"
42      client = MongoClient('localhost', port=27017, username='', password='')
43      db = client['Blockchain']
44      col = db['Bloques']
45
46      #Se guarda en la variable "colvacia" si la coleccion "Bloques" esta vacia
47      colvacia = (col.count() == 0)
48
49      #Si "colvacia" es "true" el HashAnterior es "Genesis" y el NumeroBloque es 1
50      if(colvacia == True):
51          HashAnterior = "Genesis"
52          NumeroBloque = 1
53          NumeroBloque = str(NumeroBloque)
```

Luego de evaluar si la base de datos está vacía se continúa con un condicional que elija cuando la variable “colvacia” sea falsa, esto quiere decir que la colección

“bloques” tiene información, entonces se realiza una búsqueda con “.sort” para que quede organizada de último bloque al más nuevo y con la caracteristica .limit(1) para que solo traiga 1 documento, lo anterior quiere decir que esa búsqueda traerá solo el último bloque guardado en la colección. Luego se recorre ese bloque y se almacena la información en una variable para posteriormente seleccionar los campos de “NumeroBloque” y “HashActual” que está en el bloque anterior para luego sumarle 1 a el “NumeroBloque” y convertir las 2 variables en string para luego subir a la base de datos. En la Figura 28 se muestra lo explicado en el párrafo anterior.

Figura 28. Verifica si el documento que se va a guardar no es el 1ro que estará dentro de la colección “Bloques” para luego llamar el hash del bloque anterior

```
if (colvacia == False):
    #Se guarda en la variable "bloquecodificado" el ultimo bloque que se inserto
    bloquecodificado = col.find().sort('NumeroBloque', -1).limit(1)
    #se recorren los items dentro de ese ultimo bloque
    #cada item se guarda en la variable "bloqueanterior"
    for item in bloquecodificado :
        bloqueanterior = item
    #Dentro de la lista que se guardo en "bloqueanterior" se selecciona el campo "NumeroBloque"
    numbloqant = bloqueanterior['NumeroBloque']
    #Dentro de la lista que se guardo en "bloqueanterior" se selecciona el campo "HashActual"
    hashbloqant = bloqueanterior['HashActual']
    numbloqant= int(numbloqant)
    NumeroBloque = numbloqant + 1
    HashAnterior = hashbloqant
    #Se convierten las variables en string para poderla guardar en la base de datos
    NumeroBloque = str(NumeroBloque)
    HashAnterior = str(HashAnterior)
```

En la Figura 29 se muestra una parte de una factura electrónica en XML, se muestran las etiquetas que guarda la fecha de creación de la factura electrónica y otras variables, con este ejemplo se puede entender cómo se extrae la información de esas etiquetas.

Figura 29. Estructura de un archivo XML

```
<?xml version="1.0" encoding="UTF-8"?>
- <factura>
  - <fac category="Informacion">
      <numfactura lang="en">131323123</numfactura>
      <startdate>2021-03-15</startdate>
      <enddate>2021-03-15</enddate>
      <hora>15:30pm</hora>
      <nitempresa>890321111</nitempresa>
      <nombreentidad>SURI</nombreentidad>
      <direccionentidad>Cra 63 Nro. 49 A 31. p1 Edificio Camacol</direccionentidad>
      <departamentoentidad>VALLE DEL CAUCA</departamentoentidad>
      <municipioentidad>CALI</municipioentidad>
      <nombrepaciente>Sebastian Guevara Sanchez</nombrepaciente>
      <telefono>3140004444</telefono>
      <correo>sebastrabajo96@gmail.com</correo>
      <direccionpaciente>carrera 5 calle 5 25</direccionpaciente>
      <municipiopaciente>CALI</municipiopaciente>
      <subprecio>100000</subprecio>
      <iva>12000</iva>
      <totalvalor>112000</totalvalor>
      <CUFE>12345678901235678901234567890 12</CUFE>
    </fac>
</factura>
```

4.5 ENCRIPTACIÓN DE LA INFORMACIÓN

La encriptación de información se hará en SHA256 ya que es uno de los protocolos de encriptación más difíciles de descifrar en la actualidad. Para empezar con la encriptación de la información lo primero que se hace es importar la librería hashlib que contiene varios protocolos de encriptación y entre ellos está el SHA256. Con esta librería es muy sencillo encriptar grandes cantidades de información. En la Figura 30 se muestra cómo se importa la librería hashlib.

Figura 30. Importar librerías de encriptación

```
9    import hashlib
```

Posterior a importar la librería se agrupa las variables que se encriptan, las cuales son número específico de factura, el código CUFE de la factura, la fecha de creación de la factura, el NIT de la empresa que emite la factura, el valor de la factura y por último el Hash del bloque anterior. Luego de agrupar las variables se hace llamado a la función haslib.sha256 para encriptar toda la información que se describió anteriormente en un hash creado por SHA256 y guardar el hash criptográfico en una variable que en este caso llamo “encri”. Después de guardar la variable en “encri” se

procede a convertir la variable del hash en un hexadecimal de 32 bits para poder manejar el hash de esta forma. En la Figura 31 se muestra como en la línea 58 y 59 se encriptan los datos y en la línea 61 se convierte el hash en hexadecimal y se almacena en la variable “HashActual”.

Figura 31. Encriptación de toda la información

```
58    encri= hashlib.sha256(NumFactura.encode()+CUFE.encode()+FechaCreacion.encode()
59    +NIT.encode()+ValorFactura.encode()+HashAnterior.encode())
60
61    HashActual = encri.hexdigest()
```

4.6 FUNCIONAMIENTO DE BASE DE DATOS MONGODB

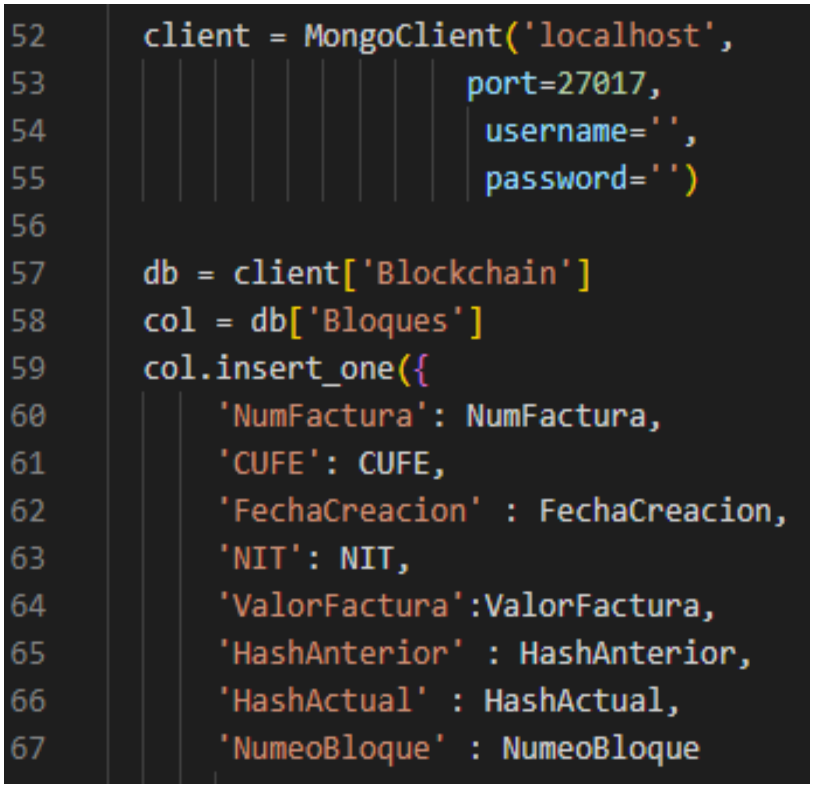
El motor de base de datos que se usa para este proyecto es MongoDB y se conectan las interfaces de usuario a la base de datos con el lenguaje Python, lo primero que se hace en el código de Python es importar las librerías de PyMongo que se usarán para conectarse, ingresar y solicitar información a la base de datos. En la Figura 32 se muestra qué librerías se usó y cómo se importan en el código.

Figura 32. Importar las librerías de pymongo en el código Python para comunicarse con el motor de base de datos

```
11    from pymongo import MongoClient
12    from pymongo import ASCENDING, DESCENDING
```

Luego de importar las librerías se procede a ingresar los datos del cliente del motor de base de datos en los cuales está “localhost”, el puerto por donde se comunicará, el nombre del usuario y su contraseña, en la Figura 33 se muestra en las líneas de código de 52 al 55. Luego de crear el código de los datos del cliente en la línea 57 se ingresa en una variable el nombre de la base de datos y luego en la línea 58 el nombre de la colección que en base de datos SQL se conoce como tabla. En la Figura 33 de la línea 59 a la 67 se muestra como las variables que tienen información extraída de la factura electrónica son ingresadas a la colección “Bloques” de la base de datos “Blockchain”.

Figura 33. Implementación del código Python para subir la factura electrónica XML



```
client = MongoClient('localhost',
                     port=27017,
                     username='',
                     password='')

db = client['Blockchain']
col = db['Bloques']
col.insert_one({
    'NumFactura': NumFactura,
    'CUFE': CUFE,
    'FechaCreacion' : FechaCreacion,
    'NIT': NIT,
    'ValorFactura':ValorFactura,
    'HashAnterior' : HashAnterior,
    'HashActual' : HashActual,
    'NumeoBloque' : NumeoBloque
```

4.7 PRUEBAS DE SOFTWARE

Las dos pruebas que se hacen son para confirmar la seguridad en el Blockchain implementado para almacenar facturación electrónica. En la Figura 34 se ve el almacenamiento de una factura electrónica llamada “Factura 1” la cual tiene el HashAnterior guardado como “Genesis” porque este es el primer bloque que se crea, también podemos notar que el código CUFE que tiene esta factura termina en el número cero.

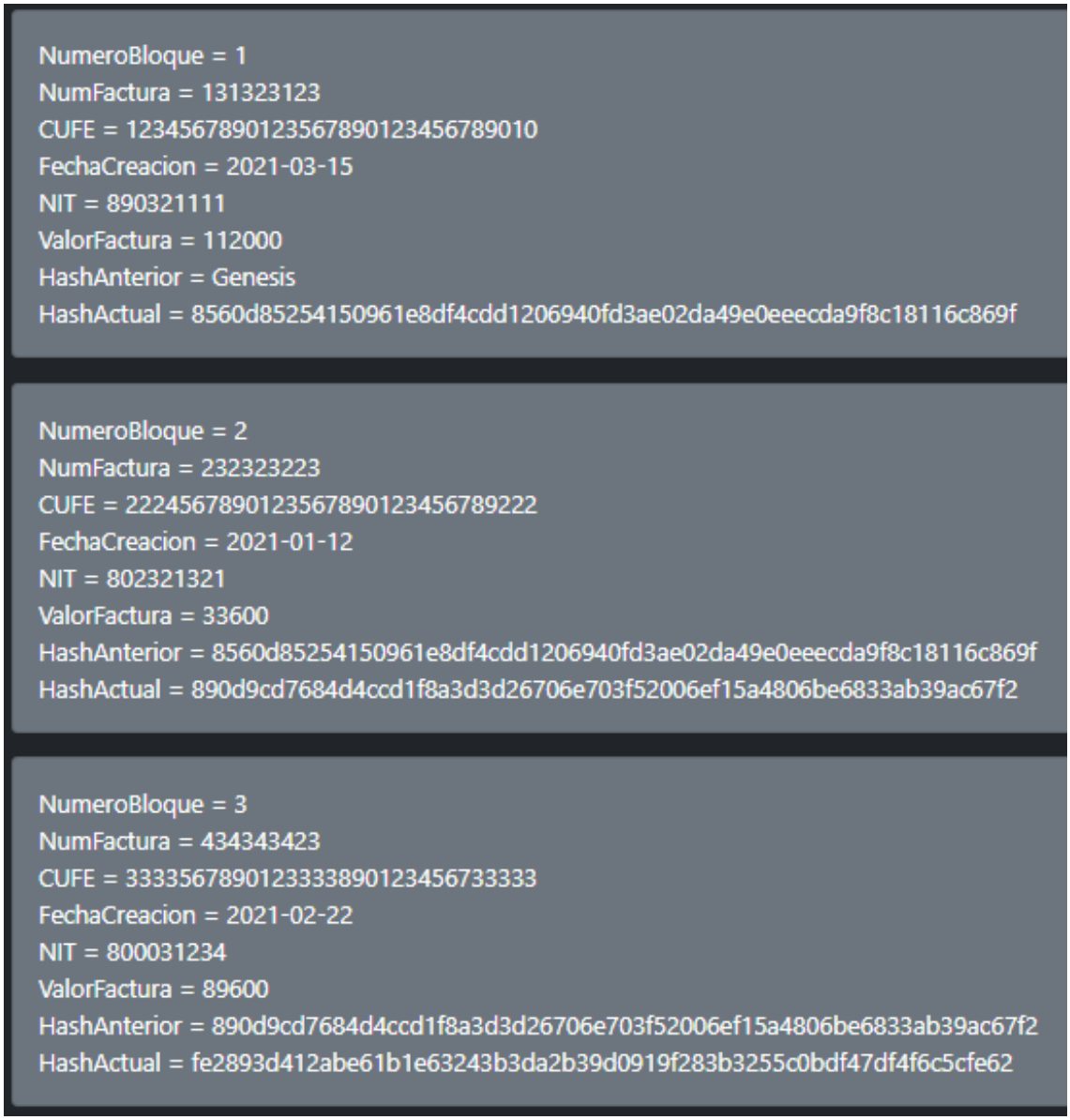
Figura 34. Primer bloque creado con la “Factura 1”

Para verificar la variación del hash se cambia en la variable del “CUFE” el último número de este código, del valor 0 al valor 9, en la Figura 35 se puede confirmar como el resto de información de la factura electrónica es idéntica a la Figura 34, con la diferencia de que cambia totalmente el HashActual del bloque creado.

Figura 35. Segundo bloque creado con la “Factura 1” con 1 dato cambiado

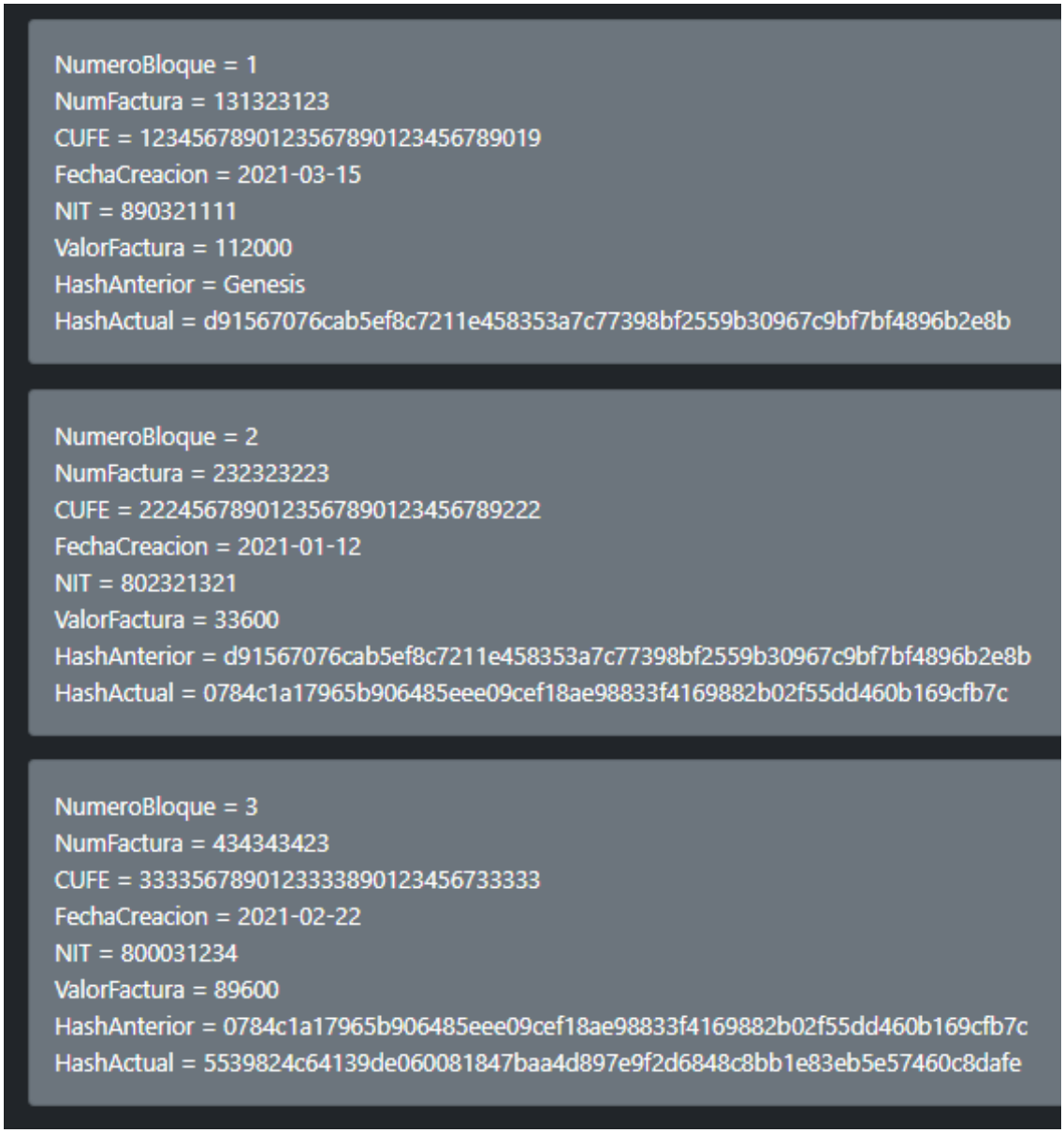
En la siguiente prueba se crea una cadena de bloques con la información de 3 facturas electrónica como se muestra en la Figura 36, observando los hashanteriores y hashactuales. Luego se crea una nueva cadena variando uno de los datos del primer bloque para comprobar la variación de hashes intermedios verificando con esto la sensibilidad en la seguridad del sistema ante intrusiones en alguno de los bloques.

Figura 36. Primer Blockchain creado con las facturas sin cambios en su información



En la Figura 37 se muestra cómo al cambiar un dato, los siguientes cadena de hash van cambiando sus datos, teniendo un hashactual diferente el bloque 3, así en la Figura 36 se muestra que el bloque 3 tiene la misma información que el bloque 3 de la Figura 36, esto se da porque en el 1er bloque fue creado con un dato diferente , esto comprueba que los hash de encriptación cambian totalmente teniendo en cuenta la información anterior de cada bloque. Esto comprueba que la seguridad en la facturación electrónica mejora con la tecnología Blockchain ya que, si cambia 1 dato en la cadena de bloques, cambia totalmente el último hashactual, y esto demuestra que la información se puede confirmar evaluando el último hash criptográfico.

Figura 37 Segundo Blockchain creado con las facturas con 1 cambio en la información



5. CONCLUSIONES

La revisión de la literatura permitió determinar que las características de un software de facturación electrónica con tecnología Blockchain son: transparencia, tecnología descentralizada, mejor seguridad, registros distribuidos y acuerdos más rápidos. Además, dentro de las herramientas usadas para este tipo de implementaciones destacan los implementadores de nodos Blockchain, compiladores de lenguajes de programación usados para Blockchain, creadores de contratos Blockchain y software para hacer pruebas de programas descentralizados. La implementación de este tipo de software se ejecuta en la mayoría de los casos usando lenguajes de programación como C++ , Solidity, Java, Python y Javascript.

Se propuso una metodología para implementar la tecnología Blockchain en un software para el almacenamiento de facturas electrónicas teniendo en cuenta los requisitos propuestos por la DIAN al momento de emitir facturas electrónicas.

La encriptación SHA-1 usada en la facturación electrónica de Colombia presenta vulnerabilidad de seguridad ante ataques de colisión de hash por su bajo nivel de encriptación. Sin embargo, el país ha mejorado su seguridad en las facturas electrónicas usando SHA-358 pero las facturas electrónicas antiguas siguen estando encriptadas en SHA-1, lo que genera una posibilidad de intrusión.

Se desarrolló un Software de facturación electrónica con tecnología Blockchain para disminuir la inseguridad en el sistema de facturación electrónica en Colombia, el cual es escalable y puede manejar gran cantidad de información.

Se validó el funcionamiento del software propuesto por medio de 2 pruebas que demuestran la seguridad en la creación de la cadena de bloques. La primera prueba demostró el cambio en el Hash del bloque ante una variación de la información interna. Mientras que la segunda reflejó la modificación de los Hash de una serie de bloques ante la alteración de 1 solo dato. Lo anterior muestra la robustez de un sistema de facturación que utiliza Blockchain como sistema de seguridad de la información.

6. RECOMENDACIONES

Al procesar gran volumen de información se recomienda descentralizar la base de datos y almacenar la información en diferentes lugares al mismo tiempo.

Cada empresa maneja su estructura de XML en su factura electrónica, se recomienda hacerle cambios al momento de extraer la información de la factura electrónica de cada empresa o de lo contrario proponer una plantilla XML propia para el uso de este software.

REFERENCIAS

[1] A. Beltran. Implementación Facturación electrónica en Colombia. Colombia: Universidad Catolica de Colombia, 2018.

[2] Addbot (2015 Nov 20) Secure Hash Algorithm [Online]. Disponible: https://acortar.link/UofnK

[3] Blockchain Es (2020, Dic 16). Guía para principiantes de la programación en Blockchain [Online]. Disponible: https://acortar.link/ATl9z

[4] CEF digital connecting Europe. (2020 Mar 24). Estonia sets an example in e-invoicing [Online]. Disponible: https://acortar.link/VwdvJ

[5] Corda (2021) Build permissioned distributed solutions and networks. [Online]. Disponible: https://www.corda.net/samples

[6] C. Nevile (2021 May 1) Enterprise ethereum alliance client specification v7. [Online]. Disponible: https://entethalliance.github.io/client-spec/spec.html

[7] D. Blanco (2021) Corda: una dlt de entidades financieras. [Online]. Disponible: https://acortar.link/tCA3M

[8] Developers circle (2021) Circle APIs Documentation. [Online]. Disponible: https://developers.circle.com

[9] Diariobitcoin (2019). Qué es una cadena de bloques / Blockchain. [Online]. Disponible: https://www.diariobitcoin.com/glossary/blockchain-2/

[10] Docuten. (2019 Nov 7).Docuten y su ecosistema Blockchain. [Online]. Disponible: https://docuten.com/es/docuten-ecosistema-blockchain/

[11] F. Guezengar (2020). Deploy & run transactions in the Blockchain.[Online]. Disponible: https://remix-project.org/

[12] Fisco Bcos (2020). Fisco Bcos. [Online]. Disponible: http://www.fisco-bcos.org/

[13] Garcia, A. (2017). El cifrado SHA-1 ya no es seguro: Google lo ha roto después de 22 años. [Online]. Disponible: https://acortar.link/IUsfg

[14] Gosocket (2020). Gosocket firma el acuerdo con el proyecto Hive para expandir el uso de Blockchain. [Online]. Disponible: https://acortar.link/XtI64

[15] hyperledger (2021) Advancing business blockchain adoption through global open source collaboration [Online]. Disponible: https://www.hyperledger.org/

[16] Isotools (2018 Jul 12) Industria 4.0, ¿qué debemos saber? [Online]. Disponible: https://acortar.link/0uiJB

[17] J. Maldonado (2018, Dic 15). Los 5 lenguajes de programación más usados en proyectos blockchain [Online]. Disponible: https://acortar.link/bs21B

[18] J. Sanchez. Técnicas de encriptación para mejorar la seguridad en la transferencia de archivos en un entorno fiable. Perú: Universidad Señor de Sipán, 2017.

[19] Kiwiz (2018). Pourquoi utiliser la blockchain pour certifier les factures de votre site e-commerce [Online]. Disponible: https://acortar.link/FmRRR

[20] LaRepublica (2019, 15 de octubre). En Colombia ya hay más de 13000 empresas que facturan electrónicamente [Online]. Disponible: https://acortar.link/fx3L8

[21] L. Barrenechea. Modelo de negocio para un sistema de seguridad, respaldo y verificación digital basado en Blockchain para la gestión documental de notarias en lima moderna. Perú: Universidad ESAN, 2020.

[22] L. Cano. Blockchain innovación como ventaja competitiva en colombia. Colombia: Universidad Cooperativa de Colombia, 2020.

[23] M. Franchi. Algoritmos de encriptación de clave asimétrica. Argentina: Universidad Nacional de la Plata, 2012.

[24] M. Fuentes (2019 Nov 7) Docuten y su tecnología Blockchain [Online]. Disponible: https://docuten.com/es/blog/docuten-ecosistema-blockchain/

[25] N. Rodriguez (2020 Sep 13). Lista de la 10 Mejores Herramientas de Blockchain [Online]. Disponible: https://n9.cl/s67u2

[26] Parity (2020, Dic 16). Blockchain Infrastructure for the Decentralised Web [Online]. Disponible: https://www.parity.io/

[27] Pymas (2020). El futuro de la facturación electrónica en Colombia. [Online]. Disponible: https://acortar.link/MJrpY

[28] R.Naik (2013 Sept 2) Optimising the SHA256 hashing algorithm for faster and more efficient bitcoin mining [Online]. Disponible: https://acortar.link/0EiHJ

[29] S. Martínez (2019) Informe final de resultados prototipo Blockchain [Online]. Disponible: https://acortar.link/m9h0l

[30] S. Nakamoto (2008). Bitcoin: Un Sistema de Efectivo Electrónico Usuario-a-Usuario [Online]. Disponible: https://acortar.link/fybWv

[31] T. Birch (2019 Oct 10). Blockchain in Tax - Electronic Invoicing [Online]. Disponible: https://thesuite.pwc.com/insights/blockchain-in-tax-electronic-invoicing

[32] Tencent (2019 May 13). China debutó con la primera factura electrónica de blockchain [Online]. Disponible: https://acortar.link/viGfS

[33] Truffle (2020 Dic 16). Sweet tools for smart contracts. Disponible: https://www.trufflesuite.com/docs/truffle/overview