Universidad de La Habana
Facultad de Matemática y Computación

Sistema de Reputación en *Blockchain* utilizando datos *on-chain*.

Autor:
José Miguel Zayas Pérez

Tutores:

Lic. Ariel Díaz Pérez
Dr. Miguel Katrib Mora

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

github.com/josem-nex/blockchain-reputation

Agradecimientos

Agredecer a todos los que de una forma u otra me han apoyado en este camino tan largo.

A mis tutores, Katrib y Ariel, por su apoyo y guía.

A mis compañeros de la facultad, por los momentos compartidos y el apoyo mutuo.

A mis padres, mi hermano y mi familia en general.

Opinión del tutor

En calidad de tutor del trabajo de diploma titulado *"Sistema de Reputación en Blockchain utilizando datos on-chain"*, elaborado por el estudiante José Miguel Zayas Pérez, deseo expresar mi criterio favorable y mi reconocimiento al excelente desempeño mostrado durante todo el desarrollo del proyecto.

Este trabajo aborda un problema de alta relevancia en el ámbito de las tecnologías descentralizadas: la necesidad de construir mecanismos de confianza nativos en entornos *blockchain*, específicamente en escenarios *peer-to-peer* (P2P). La propuesta desarrollada por el estudiante no solo identifica de forma clara los desafíos actuales de los sistemas de reputación tradicionales y centralizados, sino que además introduce una solución original, funcional y bien fundamentada. Su aporte más notable radica en el diseño e implementación de un proveedor de datos *on-chain* que permite, de forma eficiente, la extracción de métricas objetivas a partir del historial transaccional de un *wallet*, complementado con un sistema de reputación basado en contratos inteligentes que actúan como capa de caché y persistencia.

Desde el punto de vista científico y técnico, la tesis combina rigurosamente la investigación del estado del arte con una propuesta innovadora, sólida y bien implementada. El trabajo demuestra un dominio avanzado de conceptos clave de *blockchain*, contratos inteligentes, desarrollo backend y análisis de datos, integrados mediante tecnologías modernas como `Python`, `FastAPI`, `Web3.py` y `Solidity`. La arquitectura diseñada no solo es escalable y replicable, sino que contribuye al campo con una herramienta versátil, abierta y extensible que podría ser adoptada por desarrolladores de *dApps* en múltiples redes compatibles con EVM.

La actitud del estudiante durante todo el proceso fue proactiva, reflexiva y comprometida con la excelencia. Mostró una gran capacidad de investigación, creatividad en la resolución de problemas y rigurosidad en la validación experimental. Su autonomía intelectual, junto con la calidad del software desarrollado y la claridad expositiva del documento, constituyen méritos indiscutibles que avalan su idoneidad para recibir el título de Licenciado en Ciencia de la Computación. Por todo lo anterior, considero que esta tesis representa un aporte valioso al campo de estudio y que el estudiante ha alcanzado con creces las competencias requeridas para graduarse de la Universidad y sugiero al tribunal la máxima calificación.

Resumen

Cuantificar la confianza en ecosistemas descentralizados como la Blockchain constituye un desafío fundamental. Este trabajo presenta un proveedor de datos de actividad on-chain que procesa el historial transaccional de una wallet en redes compatibles con EVM (Ethereum Virtual Machine) para extraer un conjunto de métricas objetivas, sentando las bases para la creación de sistemas de reputación versátiles. La propuesta introduce el uso de un contrato inteligente como capa de persistencia y caché on-chain, que optimiza drásticamente el rendimiento en análisis subsecuentes. Su aplicabilidad se demuestra mediante el desarrollo de un sistema prototipo para mercados P2P, el cual, a través de un modelo de normalización y ponderación, traduce las métricas en una puntuación de confianza intuitiva. Los resultados, validados mediante arquetipos teóricos y pruebas de rendimiento, confirman que el modelo diferencia eficazmente perfiles de usuario y que el mecanismo de caché reduce significativamente los tiempos de análisis, validando así la viabilidad y utilidad de la solución.

Palabras Claves: Sistema de Reputación, Blockchain, On-chain, Contratos Inteligentes, Confianza Descentralizada.

Abstract

Quantifying trust in decentralized ecosystems such as the Blockchain constitutes a fundamental challenge. This work introduces an on-chain activity data provider that processes the transactional history of a wallet on EVM-compatible (Ethereum Virtual Machine) networks to extract a set of objective metrics, laying the groundwork for versatile reputation systems. The proposal introduces the use of a smart contract as an on-chain persistence and cache layer, which drastically optimizes performance in subsequent analyses. Its applicability is demonstrated through the development of a prototype system for P2P markets, which, through a normalization and weighting model, translates these metrics into an intuitive trust score. The results, validated through theoretical archetypes and performance tests, confirm that the model effectively differentiates user profiles and that the cache mechanism significantly reduces analysis times, thus validating the solution's feasibility and utility.

Keywords: Reputation System, Blockchain, On-chain, Smart Contracts, Decentralized Trust.

Índice general

Índice de figuras

Índice de tablas

Ejemplos de código

Introducción

Desde los orígenes de las sociedades humanas la **reputación** es una cualidad fundamental. Más allá del significado monetario que pueda tener, ha funcionado como un sistema alternativo para el intercambio, pues expresa el honor, la posición social y la confianza entre los individuos. Históricamente, la reputación se consideraba de modo subjetivo y difuso, difícil de cuantificar de forma precisa. Esta falta de precisión no era un error de la época, sino una característica fundamental: calcular era sinónimo de malicia y de confabulación [1]. Cuantificar la reputación abría la puerta a juicios interesados y manipulaciones ocultas, pues la necesidad de un valor numérico convertía cada gesto en una moneda de cambio sujeta a pactos y traiciones.

La llegada de la modernidad y, posteriormente, la revolución digital transformaron radicalmente este paradigma. Con el surgimiento del correo electrónico y luego de Internet, la interacción humana ha ido migrando a plataformas virtuales donde la confianza entre desconocidos se ha vuelto crítica para el comercio y la colaboración. Plataformas como eBay [2], Amazon [3] o Airbnb [4] han introducido sistemas de reputación basados en puntuaciones y reseñas, tratando de estandarizar la valoración de usuarios, productos y servicios. Estos mecanismos pretenden no solo generar un ambiente de relativa seguridad para las transacciones, sino también establecer parámetros cuantitativos que regulen la conducta y aumenten la confianza en las interacciones comerciales. Sin embargo, estos modelos, centralizados en manos de corporaciones, presentan vulnerabilidades (reseñas falsas), falta de transparencia en los algoritmos y riesgos para la privacidad. Al igual que en los primeros intentos de la Antigüedad por cuantificar el honor, estos sistemas modernos enfrentan tensiones entre la objetividad matemática y la subjetividad inherente a las relaciones humanas.

En este escenario, la tecnología *Blockchain*, surgida en 2008 con Bitcoin [5] y potenciada con los contratos inteligentes en Ethereum [6], ofrece una posible solución a los problemas mencionados. Gracias a su descentralización, inmutabilidad, trazabilidad, transparencia, autonomía y anonimato, se ha empleado para crear y gestionar sistemas de reputación de servicios externos, donde las valoraciones se establecen y verifican por consenso colectivo y a la vez son resistentes a la alteración.

Sin embargo, los sistemas de reputación basados en *blockchain* que se han encontrado en la bibliografía trasladan métricas de plataformas externas al entorno de la

blockchain, usándola solo como herramienta de almacenamiento y gestión. Si bien esto resuelve algunos problemas de los sistemas externos, no explota el verdadero potencial de la reputación para fortalecer la propia infraestructura *blockchain*. En particular, las relaciones mercantiles *peer-to-peer* (P2P) en ecosistemas descentralizados carecen de mecanismos nativos para evaluar riesgos entre usuarios, que pueden ser anónimos, lo que las hace vulnerables a estafas, engaños y limita su adopción masiva.

En la presente tesis se propone el desarrollo de un proveedor de datos de actividad *on-chain*, un servicio fundamental que procesa el historial crudo de un usuario en la red y extrae un conjunto de métricas cuantitativas y objetivas actúando como una capa base de inteligencia sobre la *blockchain*. Además, como caso de uso concreto para demostrar la utilidad de dichos datos, se construye sobre esta base un sistema de reputación prototipo diseñado para mercados P2P. Este sistema traduce las métricas en una puntuación de confianza consistente y fácil de interpretar, ofreciendo información contextual para evaluar contrapartes en transacciones sin intermediarios.

Por tanto, el **problema de investigación** de este trabajo es: ¿Cómo diseñar un sistema que procese y estandarice la actividad histórica *on-chain* de un usuario para generar métricas relevantes y garantizar su consistencia y disponibilidad en la *blockchain* sin comprometer la privacidad inherente a esta tecnología, facilitando a su vez la construcción de sistemas de reputación para casos de uso específicos?

Para intentar resolver este problema, se define como **objetivo general:** Diseñar e implementar un sistema que analice la actividad *on-chain* en redes compatibles con EVM (Ethereum Virtual Machine) para extraer métricas relevantes, y demostrar su aplicabilidad mediante la construcción de un sistema de reputación prototipo que evalúe la confiabilidad de los usuarios en un mercado P2P.

A partir de este objetivo general este trabajo de diploma se plantea los siguientes **objetivos específicos**:

- Investigar el estado del arte de los sistemas de reputación para formalizar métricas relevantes en el contexto *blockchain*, priorizando datos internos verificables.
- Diseñar un modelo para la extracción eficiente de métricas y un algoritmo que las utilice para generar una puntuación de reputación.
- Esbozar una implementación que utilice un *smart contract* en Ethereum como capa de persistencia y caché *on-chain* que permite la verificación pública de los datos.

Estructura del presente Trabajo de Diploma

Además de la presente introducción, este documento se organiza en cuatro capítulos principales y un apartado final de conclusiones y recomendaciones.

En el **Capítulo 1** se realiza un estudio exhaustivo del estado del arte de los sistemas de reputación. Se analizan los modelos centralizados tradicionales, los sistemas basados en blockchain que evalúan servicios externos y, finalmente, las plataformas emergentes que, como este trabajo, se centran en generar reputación a partir de la propia actividad *on-chain*.

En el **Capítulo 2** se presenta la Propuesta, el núcleo conceptual de la tesis. Se detalla la arquitectura del proveedor de datos de actividad on-chain, formalizando el conjunto de métricas cuantitativas que extrae del historial de una *wallet*. Se expone el mecanismo clave de optimización: el uso de un contrato inteligente como capa de persistencia y caché. Adicionalmente, se define el caso de uso ilustrativo: un sistema de reputación para mercados P2P, describiendo el proceso de normalización y ponderación para calcular una puntuación final.

El **Capítulo 3** se centra en la Implementación y Experimentos. Se describen las tecnologías (Python, FastAPI, Web3.py) y la arquitectura de software del prototipo. Se muestra la validación experimental del mecanismo de caché, donde se mide y compara el rendimiento del sistema con y sin el caché *on-chain* para demostrar su mejora en la eficiencia.

En el **Capítulo 4** se presentan los Resultados del Sistema de Reputación a través de un análisis de casos teóricos. Se definen cinco arquetipos de usuario y se aplica el modelo de reputación para evaluar su capacidad de diferenciar comportamientos y asignar puntuaciones lógicas, validando así la coherencia y utilidad del caso de uso propuesto.

Por último, en las Conclusiones se recogen los principales hallazgos, se valora el grado en que el sistema cumple los objetivos planteados y se discuten sus limitaciones. En el apartado de Recomendaciones se sugieren líneas de continuidad para ampliar el trabajo, como el refinamiento de métricas, la extensión a otras *blockchains* o la optimización utilizando herramientas más poderosas.

Capítulo 1

Estado del Arte

1.1. Sistemas de Reputación tradicionales y centralizados.

En las últimas dos décadas, las plataformas centralizadas de economía digital han incorporado mecanismos de reputación basados en el feedback de los usuarios para con ello fomentar la confianza entre desconocidos y reducir la asimetría de información. Los sistemas de reputación tradicionales y centralizados se basan en que los usuarios otorguen valoraciones (ya sean estrellas, votos o puntuaciones en distintas métricas) y, opcionalmente, comentarios de texto. Estas calificaciones se muestran en el perfil del vendedor, del producto o del anfitrión, generando un índice de confianza que otros usuarios consultan antes de decidirse a comprar, reservar o interactuar.

1.1.1. eBay

La plataforma eBay surgió a mediado de los años noventa como un sitio de subastas y ventas entre particulares, presenta un sistema de reputación basada en un *feedback* bilateral: comprador y vendedor valoran mutuamente la transacción como positiva, neutral o negativa, además de un comentario textual opcional (modelo similar al que hoy día aplican sistemas con Airbnb y Uber). El sistema acumula en el perfil de cada miembro la suma de las calificaciones de cada tipo recibidas durante los últimos seis meses.

Este sistema es sencillo de entender y permite evaluar rápidamente a cada contraparte, sin embargo estudios [2] demuestran que menos del 1 % de las calificaciones totales son negativas, convirtiéndolo prácticamente en un sistema binario donde la mayoría de los usuarios valoran la transacción como positivo o deciden no publicar una valoración, dificultando discernir entre vendedores realmente confiables y aquellos que reciben pocos votos negativos simplemente por inacción de los compradores,

pues eBay no muestra cuántas de sus ventas no recibieron ningún voto, lo que impide calcular proporciones de *feedback* fiables.

1.1.2. Amazon

Amazon comenzó como una librería *online* y evolucionó rápidamente hacia un mercado mixto: vende productos y aloja miles de vendedores externos. Su sistema de reputación se centra en calificaciones de 1 a 5 estrellas y reseñas de texto, complementadas con votos de útil otorgados por otros usuarios permitiéndole destacar las reseñas más relevantes. En Amazon, a diferencia de eBay, la reputación es un indicador de la calidad de cada producto más que del vendedor en sí, aunque al final la confiabilidad del vendedor también se refleja en comentarios y estadísticas de puntaje. No todos los usuarios tienen permitido emitir puntuaciones y reseñas, solo pueden hacerlo aquellos que han gastado al menos $50 dólares en la plataforma en los últimos 12 meses [7], lo que ayuda a amortiguar el problema de reseñas falsas.

A pesar de las ventajas ya mencionadas, Amazon introdujo nuevas limitaciones como los sesgos de acumulación: los productos con muchas reseñas positivas tienden a recibir más atención y más reseñas, dificultando que productos nuevos ganen reputación o destaquen. A su vez, la mayor complejidad y manipulación, tanto de las puntuaciones como de las reseñas, comenzaron a hacer evidente uno de los grandes problemas de la centralización, la falta de transparencia. Amazon no revela cómo pondera las puntuaciones útiles o como calcula el *ranking* de reseñas, impidiendo así un análisis real del impacto de cada reseña.

1.1.3. Airbnb

Airbnb permite a anfitriones y huéspedes reservar y rentar espacios en hogares particulares, estableciendo relaciones directas entre desconocidos. Al finalizar cada estancia, los huéspedes califican al anfitrión según varias métricas (limpieza, comunicación, etc.) y pueden añadir un comentario. Aunque los anfitriones también pueden responder, la plataforma prioriza la valoración del huésped para determinar la visibilidad del anuncio. Airbnb además incorpora datos internos (rapidez de respuesta a mensajes, tasa de aceptación de reservas, etc) en un algoritmo que determina la posición de cada anfitrión en los resultados de búsqueda. Este sistema de reputación busca aumentar la información de las ofertas en un entorno tan vulnerable como es el alojamiento y la obligación de pagar por adelantado.

Aunque esta estructura ha permitido generar cierta confianza entre desconocidos, también introduce una serie de inconvenientes que afectan principalmente a los anfitriones, especialmente a los de menor puntuación [4]. Algunas de ellas son: el uso de métricas predeterminadas que priorizan ciertos aspectos e invisibilizan otras dimen-

siones de la experiencia; la opacidad del algoritmo, que impide comprender cómo se ponderan calificaciones y datos internos; y la presencia de huéspedes con estándares extremadamente estrictos o con prejuicios raciales y de clase, que asignan valoraciones bajas injustamente, distorsionando la reputación real del anfitrión.

1.1.4. Desafíos generales de los sistemas tradicionales

La mayoría de los sistemas de reputación tradicionales, como los anteriormente mencionados, comparten una falta de transparencia en cómo ponderan internamente cada voto o métrica, lo que impide un análisis real de la relevancia de cada opinión y favorece sesgos como la sobrevaloración general (pocas reseñas negativas) o la acumulación de popularidad en productos ya consolidados.

El principal problema que enfrentan estas plataformas es la verificación de la autenticidad de las reseñas o puntuaciones. Aunque algunas limitan quién puede dejar *feedback* o basan parte del algoritmo en métricas internas, siguen existiendo prácticas de reseñas falsas o *review bombing* (personas que se organizan para publicar masivamente reseñas negativas a un producto o servicio) y omisión de valoraciones negativas por parte de usuarios que prefieren no participar. Esta carencia de mecanismos confiables para garantizar que cada calificación proceda de una experiencia real dificulta distinguir entre verdaderos proveedores de calidad y aquellos que simplemente evitan el *feedback* desfavorable.

1.2. Sistemas de Reputación de servicios externos basados en *Blockchain*.

Si bien los Sistemas de Reputación centralizados tradicionales han sido pioneros para establecer confianza en los escenarios en línea, su dependencia de una autoridad central para la recolección, el cómputo y el almacenamiento de la reputación nos presenta ante desafíos significativos en términos de transparencia, sesgos, puntos únicos de fallo y resistencia a la manipulación [8]. La necesidad de una mayor confiabilidad en el propio sistema de gestión de la reputación ha impulsado la exploración de tecnologías alternativas.

La tecnología *Blockchain* pudiera ser útil para la implementación de sistemas de reputación que sean más transparentes, seguros y descentralizados [9]. La naturaleza inmutable y auditable de la *blockchain* ofrece un mecanismo para registrar interacciones y puntuaciones de reputación de manera que sea difícil de alterar y que a la vez esté abierta a la verificación por los participantes. Estos sistemas buscan que el concepto de confianza pase de ser controlado centralizadamente hacia un protocolo y red distribuida que permita a las partes interactuar con un mayor grado de seguridad

incluso sin un conocimiento previo mutuo profundo.

Los Sistemas de Reputación basados en *Blockchain* suelen abordar la gestión, cómputo y almacenamiento de la reputación de manera cuantitativa, facilitando la toma de decisiones sobre la confianza entre entidades. A menudo, estos sistemas emplean una combinación de almacenamiento y cómputo *on-chain* y *off-chain* para equilibrar las ventajas de la *blockchain*, con las necesidades de escalabilidad, privacidad y eficiencia de costos. La interacción con servicios externos o datos del mundo real se facilita comúnmente a través de oráculos, servicios que actúan como puentes entre el mundo *on-chain* y *off-chain* [8]. A continuación, se analizarán dos enfoques y sistemas representativos que ilustran la aplicación de *blockchain* para la gestión de la reputación de servicios externos.

1.2.1. REPUTABLE: Un Sistema de Reputación Descentralizado para Ecosistemas Basados en *Blockchain*

El sistema REPUTABLE, presentado por Arshad [10], se enfoca en calcular la reputación de proveedores de servicios externos dentro de un ecosistema *blockchain* mediante una implementación descentralizada *on-chain* y *off-chain*. Su objetivo principal es garantizar no solo la privacidad, sino también la fiabilidad, integridad y precisión de los valores de reputación con un sobrecosto mínimo.

REPUTABLE propone una arquitectura híbrida en la que el *feedback* individual de los usuarios, que es la base para el cálculo de la reputación, se almacena *off-chain* para mejorar la escalabilidad y reducir costos. Los metadatos y las puntuaciones de reputación agregadas se almacenan *on-chain* para garantizar la transparencia y la verificabilidad. La interacción entre los componentes on-chain y off-chain, especialmente para la recolección de *feedback*, se gestiona a través de un servicio de oráculo de datos.

Para garantizar la validez del *feedback* dentro de la red, REPUTABLE utiliza un sistema de generación de *tokens* (unidades digitales que representan valor o derechos en una blockchain): se emiten *tokens* únicos por cada transacción cualificada (ej. compra de un servicio), y solo sus poseedores pueden enviar *feedback*. Un aspecto central de REPUTABLE es la preservación de la privacidad del usuario. En lugar de enviar el *feedback* en texto plano, los usuarios envían criptogramas de sus valoraciones. El sistema utiliza criptografía para permitir la agregación de estos *feedbacks* encriptados sin revelar las puntuaciones individuales. Esto protege a los usuarios de posibles represalias y asegura que ni siquiera el agregador, quien realiza la agregación de la reputación utilizando los *feedbacks*, pueda conocer las valoraciones individuales. El cálculo de la reputación puede utilizar varios modelos (ej. suma, promedio) calculados a partir de las puntuaciones positivas y negativas, en el ejemplo de uso se propone BRS (Beta Reputation System [11]). Los usuarios pueden verificar públicamente las

puntuaciones de reputación calculadas. Entre los componentes claves del sistema también se encuentran los *smart contracts* utilizados para procesar y almacenar los datos *on-chain* así como interactuar con el almacenamiento *off-chain*.

1.2.2. DEM-BTRM: Modelo de Confianza y Reputación Basado en *Blockchain* con Mecanismo de Evaluación Dinámica para IoT (Internet of Things)

El modelo DEM-BTRM (Dynamic Evaluation Mechanism - Blockchain-based Trust and Reputation Model) propuesto por Tu [12] está diseñado para el Internet de las Cosas (IoT). Su objetivo es realizar una evaluación integral del comportamiento del usuario (dispositivo IoT) desde múltiples perspectivas y resistir ataques maliciosos, especialmente los nativos de redes distribuidas.

El sistema se estructura en tres capas: - Capa de Dispositivos IoT (IDL): Contiene los dispositivos IoT y usuarios (UEs). - Capa de Acceso y Reenvío (AFL): Compuesta por routers de reenvío y pasarelas de acceso. - Capa de Servicios *Blockchain* (BSL): Los servicios de red se despliegan en *smart contracts*. El modelo se implementa sobre Hyperledger Fabric (plataforma blockchain enfocada en soluciones empresariales).

DEM-BTRM evalúa el comportamiento del usuario de forma multidimensional, considerando su comportamiento dentro de la red en acciones como: autenticación, conexiones, adquisición de recursos, comunicación, entre otros. La reputación directa se calcula utilizando el Beta Reputation System mejorado con un factor de penalización para el comportamiento negativo, también se considera una reputación indirecta basada en las evaluaciones de otras pasarelas. La principal innovación es el Mecanismo de Evaluación Dinámica, que incluye un algoritmo que ajusta dinámicamente el intervalo de evaluación de la reputación. Los usuarios con alta reputación tienen intervalos más largos, mientras que aquellos con baja reputación o comportamiento sospechoso son evaluados más frecuentemente. A su vez, se implementa un Mecanismo de Decaimiento de la reputación para usuarios inactivos a largo plazo, evitando que mantengan una alta reputación indefinidamente y previniendo que otros usuarios maliciosos recuperen reputación rápidamente tras un periodo de inactividad. DEM-BTRM ofrece un enfoque sofisticado para la gestión de la reputación en IoT, destacando por su capacidad de adaptación dinámica y su evaluación comprensiva del comportamiento, lo que lo hace más resiliente a tácticas de ataque y manipulaciones.

1.2.3. Desafíos, limitaciones y soluciones de los Sistemas de Reputación basados en *Blockchain*.

A pesar de las ventajas que la tecnología *blockchain* aporta a los sistemas de reputación, y de sus casos de uso exitosos, su implementación no está exenta de

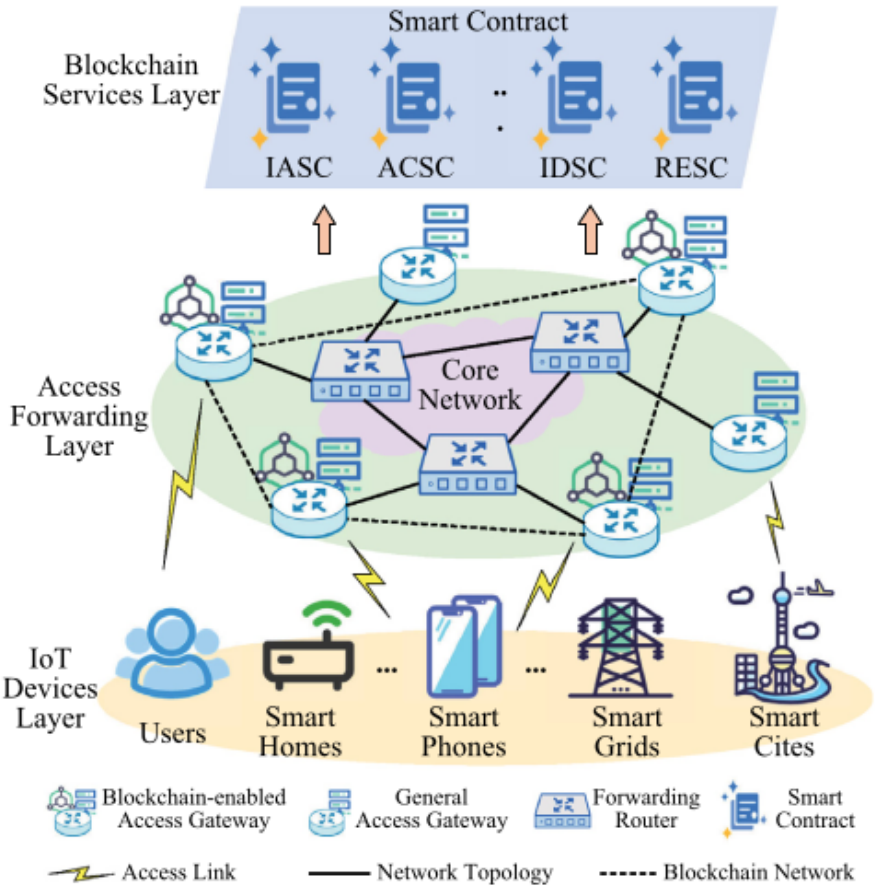

Figura 1.1: Arquitectura del BTRM (Blockchain-based Trust and Reputation Model)

desafíos y limitaciones [8, 9]. A continuación se enumeran algunos de ellos, así como las soluciones o adaptaciones que se han utilizado para mitigar estos problemas:

- Escalabilidad y costos: Guardar todos los datos de *feedback* y transacciones directamente en la *blockchain* puede ser considerablemente caro e ineficiente, llevando a un crecimiento rápido del tamaño de la red, además, *blockchains* públicas como Ethereum tienen un límite en el número de transacciones por segundo, lo que puede generar cuellos de botella y retrasos, especialmente en sistemas con alta frecuencia de interacciones o actualizaciones de reputación. Se ha conseguido mitigar este problema empleando almacenamiento *off-chain* para datos voluminosos o menos críticos y para el cómputo, registrando solo hashes o resultados agregados *on-chain*.

- Privacidad vs. transparencia: La transparencia, inherente a las *blockchains* públicas, si bien es beneficiosa para la auditoría y la verificabilidad, puede entrar en conflicto con la privacidad de los usuarios, especialmente si el *feedback* o las identidades de los evaluadores quedan expuestos. El uso de pruebas de conocimiento cero (ZKP, técnicas criptográficas para verificar información sin revelarla), criptografía, seudónimos o *blockchains* permisionadas han logrado solucionar gran parte de este problema.

- Robustez frente a ataques específicos de reputación: A pesar de las mejoras, los Sistemas de Reputación basados en *Blockchain* aún pueden ser vulnerables a ataques como Sybil (crear múltiples identidades falsas para influir en una red) o colusión *off-chain*. La mayoría de los ataques se pudieran resolver sacrificando en parte la privacidad, utilizando mecanismos de identidad más robustos, costos de entrada al sistema, o algoritmos de detección de comportamiento anómalo.

- Smart Contracts: Los *smart contracts*, a pesar de ser fundamentales para la lógica *on-chain* de los Sistemas de Reputación basados en *Blockchain*, presentan varias limitaciones intrínsecas, como son: las restricciones computacionales y costos de gas (especialmente en plataformas como Ethereum); la falta de soporte nativo para ciertas operaciones, como las de punto flotante, restringiendo la implementación directa de algunos modelos matemáticos; el determinismo, lo que impide que los *smart contracts* generen aleatoriedad de forma nativa, ya que esto podría llevar a resultados inconsistentes y finalmente, la inmutabilidad, que si bien es una característica de seguridad, plantea desafíos significativos para la actualización y la gobernanza pues una vez desplegado, modificar un *smart contract* es difícil y a menudo requiere desplegar una nueva versión y migrar los datos, lo cual puede ser un proceso complejo y costoso, decidir sobre cambios o mejoras al protocolo de reputación implementado en los *smart contracts* necesita mecanismos de consenso entre los participantes del sistema.

Si bien los Sistemas de Reputación basados en *Blockchain* ofrecen soluciones innovadoras a problemas de confianza y transparencia, es crucial un diseño cuidadoso que considere estas limitaciones y busque un equilibrio óptimo entre descentralización, seguridad, privacidad, eficiencia y usabilidad.

1.3. Sistemas de Reputación *on-chain.*

Mientras que los sistemas de reputación analizados previamente suelen establecer puentes con servicios externos, es necesario también que dentro del propio ecosistema *blockchain* se puedan construir sistemas de reputación que operen y deriven su valor directamente de la actividad registrada en la cadena. Estos sistemas *on-chain* buscan impulsar la creación de mecanismos de confianza nativos en un entorno descentralizado y seudónimo donde las garantías de identidad tradicionales son a menudo inexistentes.

La utilidad y necesidad de desarrollar sistemas de reputación puramente *on-chain* [13] radica en:

- Fomentar la confianza en Web3[1]: A medida que las interacciones en DeFi (Finanzas Descentralizadas), DAOs (Organizaciones Autónomas Descentralizadas) y NFTs (Tokens No Fungibles) se complejizan, una reputación *on-chain* verificable permite tomar decisiones más informadas y mitigar los riesgos de estafas.
- Incentivar el comportamiento positivo: Un buen historial *on-chain* puede actuar como capital social, desbloqueando mejores oportunidades y fomentando la participación constructiva.
- Aportar contexto a la data transaccional: Estos sistemas buscan estandarizar, procesar y priorizar la afluencia de datos de la *blockchain* para construir perfiles de reputación más ricos que una simple lista de transacciones.
- Estándar y credibilidad interoperable: El objetivo es crear un marco estandarizado y general para que la reputación *on-chain* sea portable a través de dApps (Aplicaciones descentralizadas) y *blockchains*.
- Aumentar la seguridad: La identificación de patrones de comportamiento malicioso directamente en la cadena puede alimentar herramientas de prevención de fraudes.

[1]Visión de una Web descentralizada basada en la *Blockchain* y en *Smart Contracts*, se usa para referirse a todo el ecosistema en su conjunto.

Dada la creciente importancia de estos mecanismos para la madurez y seguridad del ecosistema Web3, diversas iniciativas y proyectos ya están explorando y desarrollando soluciones para la reputación *on-chain*, a continuación se presentan algunos de los enfoques y sistemas más representativos.

1.3.1. Gitcoin Passport

Gitcoin es una plataforma que nació con la misión de "hacer crecer el código abierto" [14] (Grow Open Source). Originalmente, Gitcoin se estableció como un espacio en la *blockchain* de Ethereum para conectar a desarrolladores de software libre con proyectos que necesitaban contribuciones, facilitando que los desarrolladores fueran recompensados por su trabajo y que los proyectos ganaran en visibilidad a una comunidad activa. Su enfoque principal ha sido fomentar la colaboración incentivada en el desarrollo de *software* de código abierto, operando inicialmente sin un *token* propio y soportando diversos *tokens* para promover el ecosistema Ethereum en su conjunto. A partir de esta base centrada en la comunidad y la contribución, y reconociendo la necesidad de verificar la identidad de forma que se preserve la privacidad y se mitiguen los ataques Sybil, Gitcoin desarrolló Gitcoin Passport.

Gitcoin Passport se presenta como una capa de identidad para Web3 resistente a Sybil y que preserva la privacidad. Es fundamentalmente un agregador de verificación de identidad descentralizado que permite a los usuarios demostrar su humanidad única y construir una reputación *on-chain* sin comprometer su información personal identificable. Su funcionamiento se basa en la recolección de Stamps (sellos o credenciales verificables). Estos Stamps representan actividades o afiliaciones que son relativamente fáciles de verificar para un humano, pero difíciles de falsificar a gran escala por *bots* o actores maliciosos.

Los Stamps se agrupan en diversas categorías, incluyendo actividad en *blockchain* y otras redes cripto; identificaciones Gubernamentales (KYC, Know Your Client) a travez de integraciones con plataformas de comercio de criptomonedas como Binance o Coinbase; plataformas sociales y profesionales como conexiones con cuentas de Google, LinkedIn, Discord, entre otras; verificación biométrica, pruebas de unicidad y *liveness* (prueba de vida) mediante servicios como Civic, plataforma *blockchain* enfocada a la gestión seguro de identidad digital.

Los usuarios tienen la libertad de elegir qué Stamps verificar, acumulando un Unique Humanity Score (Puntuación de Humanidad Única) basado en el peso asignado a cada Stamp. Una puntuación que supera un umbral determinado (normalmente 20 puntos) sugiere que el usuario es probablemente un humano único. Esta puntuación puede ser utilizada por diversas aplicaciones y comunidades para conceder o denegar acceso a funcionalidades, recompensas o derechos de gobernanza. Crucialmente, Gitcoin Passport está diseñado para preservar la privacidad: al conectar Stamps, se crea

una credencial verificable que prueba la realización de una actividad específica, pero no recopila ni expone los datos personales subyacentes. Los Stamps además tienen un periodo de expiración (generalmente 90 días), lo que requiere una reverificación periódica para mantener la robustez del sistema contra ataques y permitir a los usuarios reutilizar credenciales con nuevas *wallets* si fuera necesario.

El caso de uso más emblemático de Gitcoin Passport ha sido su propia plataforma de Gitcoin Grants. En las rondas de financiación de proyectos de código abierto, Passport se utiliza para verificar la identidad de los donantes y así asegurar una distribución más justa de los fondos. Por ejemplo, en Gitcoin Grants Round 15, los donantes con un Trust Bonus Score (derivado de su Passport) suficientemente alto podían aumentar el impacto de sus donaciones incrementando la fiabilidad de los proyectos que apoyaban.

Más allá de Gitcoin Grants, Passport se ha diseñado para ser una herramienta de infraestructura para el ecosistema Web3 ofreciendo protección a DAOs y comunidades mediante visados de entrada personalizables que filtran *bots* en procesos de gobernanza, *airdrops*[2] o acceso a contenido. Además, facilita la construcción de una reputación transferible, donde los Stamps acumulados sirven como credenciales verificables en diversas plataformas. Gracias a su API y SDK, los desarrolladores pueden integrar Passport en cualquier dApp y *blockchain*, y las plataformas pueden diseñar programas de recompensas y lealtad que beneficien genuinamente a usuarios auténticos.

1.3.2. Octan Network

Octan Network es un proyecto de Octan Labs [15] enfocado en proporcionar análisis de datos *on-chain*, así como puntuaciones de reputación y clasificación para cuentas a través de múltiples *blockchains* y aplicaciones. La visión de Octan Network es convertirse en una firma líder en el análisis de *insights* (conocimientos profundos) de Web3, con la misión de construir un sistema de reputación descentralizado para establecer confianza y credibilidad en este ecosistema. Para ello, definen, analizan y etiquetan las interrelaciones y conexiones de las entidades Web3 a partir de datos *on-chain*, midiendo sus niveles de actividad y comportamientos para cuantificarlos de manera accesible.

El núcleo de su propuesta es el Reputation Ranking System (RRS), un motor que emplea algoritmos matemáticos de clasificación, incluyendo una adaptación del *PageRank* de Google y otros algoritmos de *ranking* por pares, para cuantificar las puntuaciones de reputación de usuarios y aplicaciones. El RRS captura las actividades y comportamientos de los usuarios a través de registros *on-chain* como transacciones, interacciones con contratos, volumen transaccional y gas gastado, proporcionando una

[2]Distribución gratuita de tokens a usuarios, generalmente por promocionar o estar en la fase inicial de proyectos.

medida universal, comparativa y cuantitativa de la reputación. La intuición detrás de su *ranking* es que una mayor cantidad de transferencias entrantes (In_degree), especialmente desde cuentas con alta reputación, incrementa la puntuación, mientras que las transferencias salientes (Out_degree) la disminuyen, considerando también factores como la antigüedad y el valor de las transacciones.

Octan Network distingue varios tipos de puntuaciones de reputación. El Global Reputation Score (GRS) mide la importancia de una cuenta en toda una *blockchain*, pudiendo luego segmentarse entre contratos y Cuentas de Propiedad Externa (EOAs), y filtrar *wallets* de *exchanges* para obtener una lista pura de usuarios finales. El Category Reputation Score (CRS) se calcula a partir de transacciones relacionadas con categorías específicas como DeFi, NFT, DAO o GameFi (fusión de videojuegos y finanzas descentralizadas) dentro de una *blockchain*. Finalmente, el Project/Dapp-based Reputation Score (PRS) mide la calidad de las cuentas que interactúan con los contratos de un proyecto específico. Esta granularidad permite diversos niveles de segmentación y filtrado para diferentes propósitos.

Las aplicaciones del Reputation Ranking System de Octan Network son principalmente orientadas a extraer valor del complejo grafo transaccional de Web3. Destaca su utilidad para la evaluación de inversiones, permitiendo a los inversores analizar la actividad de los contratos de un proyecto y la calidad de las *wallets* de sus usuarios, identificando oportunidades prometedoras más allá de las métricas puramente financieras. A su vez el RRS facilita la investigación y el análisis de tendencias emergentes al rastrear los cambios en la reputación de las entidades. En el ámbito del *marketing*, se enfoca en la adquisición y segmentación avanzada de usuarios, ofreciendo a proyectos y agencias herramientas para clasificar y cualificar audiencias de forma precisa. También sus puntuaciones pueden actuar como credenciales de identidad Web3 que verifican la actividad e influencia, y buscan impulsar el desarrollo de DeFi y DAOs al fomentar la transparencia y la responsabilidad dentro de estos sistemas.

Octan Network ha diseñado sus productos para ser interoperables, con planes de soportar una amplia gama de plataformas, priorizando inicialmente las compatibles con EVM. Octan Labs ha validado la fiabilidad de su sistema midiendo cómo sus rankings se correlacionan con otras estadísticas *on-chain*, buscando un equilibrio que demuestre utilidad más allá de simples estadísticas normalizadas, un caso de estudio es el *airdrop* del *token* ARB, donde la distribución de su GRS mostró una alta similitud con la distribución de los receptores del airdrop, sugiriendo la idoneidad de su sistema para filtrar *wallets* no elegibles.

1.3.3. Resumen de los Sistemas de Reputación *on-chain*.

Este es un campo que está en una fase de desarrollo y experimentación activa, con desafíos pendientes en cuanto a la estandarización, la adopción masiva y la sofisti-

cación de los modelos para capturar matices del comportamiento. Algunos proyectos como los que se han analizado aquí han demostrado su potencial para fomentar la seguridad, incentivar la participación positiva y aportar transparencia a las interacciones descentralizadas, marcando un paso crucial hacia la madurez de la Web3.

1.4. Proof of Reputation (PoR)

Mientras que numerosos sistemas de reputación en el ámbito *blockchain* se centran en evaluar entidades o servicios externos, o en construir perfiles de identidad *on-chain* que sirven como información adicional para la toma de decisiones de los usuarios, el Proof of Reputation (PoR) [16] integra directamente la reputación como el mecanismo fundamental para el consenso y la validación de transacciones dentro de la propia *blockchain*. En este modelo, el valor de reputación de un nodo no es simplemente un indicador para consultar, sino que es un factor decisivo que determina su capacidad para participar activamente en la creación de bloques, en la selección de líderes y en la gobernanza general del protocolo.

El funcionamiento de PoR se articula en torno a la selección de un grupo de consenso y la elección de un líder por cada ronda. Al inicio de una ronda de consenso, se forma un grupo compuesto por aquellos nodos que poseen los valores de reputación más altos y cuya reputación colectiva supera el 50 % de la reputación total de la red. El tamaño de este grupo es variable, dependiendo de la distribución de la reputación en la red en ese momento. De entre los miembros del grupo, se selecciona aleatoriamente un líder para la ronda actual, este líder es responsable de empaquetar las transacciones válidas pendientes en un nuevo bloque, calcular los nuevos valores de reputación para todos los nodos de la red basados en las interacciones de la ronda, y difundir un mensaje al grupo de consenso que incluye el bloque propuesto, la clave pública del líder, un hash del bloque y la lista actualizada de reputaciones.

Los demás miembros del grupo de consenso verifican la integridad y validez del mensaje del líder. Si un miembro considera válido el bloque y la información asociada, emite su propia verificación al grupo. Se considera que se ha alcanzado un consenso para publicar el bloque si un conjunto de nodos, cuya suma de pesos de reputación (basada en la reputación de la ronda anterior) excede un umbral de dos tercios del peso total del grupo de consenso, lo aprueba. Una característica distintiva de PoR es la gestión de los valores de reputación a través de una *sidechain* (cadena *blockchain* lateral) dedicada, vinculada a la cadena de transacciones principal. Esto asegura la transparencia y auditabilidad de las reputaciones sin depender de una entidad centralizada, donde cada bloque de reputación en la sidechain se corresponde con un bloque de transacciones en la cadena principal.

Inicialmente, todos los nodos comienzan con un valor de reputación por defecto. Tras las interacciones, los nodos se califican mutuamente (con valores entre 0 y 1).

Estas calificaciones recibidas por un nodo se normalizan para evitar valores nulos y escalarlas adecuadamente. Posteriormente, para calcular un puntaje intermedio (P), estas calificaciones normalizadas (S) se ponderan según la reputación de los nodos emisores de dichas calificaciones (R de la ronda previa). La nueva reputación para un nodo i en la ronda k+1 ($R_{i,k+1}$) se calcula entonces como una mezcla ponderada de este puntaje P y la propia reputación previa del nodo (R_k), utilizando una constante alpha que determina la prioridad de las calificaciones recientes:

$$R_{i,k+1} = \alpha \cdot P + (1 - \alpha) \cdot R_k$$

Finalmente, este valor se ajusta mediante una función sigmoide[3] para acotar las fluctuaciones y mantener los valores dentro de un rango manejable. Los resultados experimentales presentados por los autores indican que PoR puede alcanzar un alto rendimiento, logrando hasta 1,100 transacciones por segundo en una red de 1,000 nodos, superando a mecanismos como Proof-of-Work (PoW) y Proof-of-Stake (PoS) en este aspecto, aunque el tiempo de producción de bloques y de consenso aumenta con el tamaño del bloque y el volumen de transacciones.

El principal caso de uso de PoR es servir como un mecanismo de consenso alternativo, más eficiente energéticamente y potencialmente más seguro para diversas aplicaciones *blockchain*, particularmente en entornos abiertos donde la evaluación continua del comportamiento de los participantes es crucial.

1.5. Fundamentación de la Propuesta

El análisis del estado del arte revela un espectro diverso de sistemas de reputación, desde los modelos centralizados tradicionales hasta los mecanismos de consenso nativos de *blockchain* como Proof of Reputation. Cada enfoque presenta un balance particular entre transparencia, escalabilidad, privacidad y aplicabilidad. Esta revisión ha sido fundamental para delinear la propuesta de esta tesis, la cual se inspira en conceptos clave de los sistemas analizados mientras busca ofrecer una solución con un enfoque distintivo e innovador.

La principal inspiración proviene de Octan Network, adoptando su enfoque de analizar la actividad puramente *on-chain* para derivar métricas de comportamiento. Sin embargo, la presente propuesta, además de proveer un servicio para ser utilizado por terceros, integra un contrato inteligente como capa de persistencia y caché creando así una arquitectura híbrida similar a REPUTABLE. Este mecanismo almacena los

[3]Función matemática utilizada para transformar cualquier número de entrada a un valor dentro de un rango acotado. Se utiliza para normalizar datos, suavizar cambios bruscos y evitar valores extremos.

resultados agregados y el último bloque procesado, optimizando drásticamente la eficiencia en análisis subsecuentes.

Aunque tanto Gitcoin Passport como la presente propuesta se basan en añadir una capa de información portable, Passport se enfoca en la verificación de identidad y unicidad del usuario (quién es) y nuestra propuesta en evaluar su comportamiento histórico dentro de la red (cómo actúa). Dado que la tarea de verificar la identidad recae en la aplicación o plataforma que consume nuestro servicio, se decidió no realizar una integración directa con Gitcoin Passport. No obstante, dicha aplicación o plataforma sí podría utilizarlo como confirmación de humanidad de un participante y, posteriormente, emplear nuestro sistema para evaluar la fiabilidad de su historial transaccional antes de realizar una operación crítica.

Capítulo 2

Propuesta de extracción de métricas y modelado de la Reputación en la *Blockchain*

La solución que se presenta en la presente tesis se puede dividir en dos partes, primero el desarrollo de un API que, dada una blockchain compatible con EVM y a partir de la dirección de una *wallet*, pueda analizar su historial dentro de la red para extraer un conjunto de métricas relevantes. Este servicio actuará como la capa base que proporcionaría los datos de la actividad *on-chain* del usuario que serán utilizados posteriormente por sistemas de reputación. La segunda parte describe cómo utilizar las métricas para construir un sistema de reputación prototipo, este sistema servirá como un caso de uso concreto, mostrando como la información *on-chain* puede traducirse en una puntuación de reputación útil.

2.1. Proveedor de datos de actividad *on-chain*

La tecnología *blockchain* asegura que cada interacción de una cuenta se preserve de forma inmutable como una transacción, sin embargo, la interpretación de este historial de transacciones, que a menudo puede ser extenso y técnico, resulta complejo principalmente para usuarios no desarrolladores. Lo que pretendemos es crear un proveedor de datos de actividad *on-chain* que procese los datos crudos, extrayendo métricas cuantitativas que representan los aspectos claves de la actividad *on-chain*.

Para optimizar la eficiencia y robustez se utilizará un contrato inteligente que actuará como una capa de persistencia y *caché on-chain* para las métricas obtenidas. Su función principal será almacenar, para cada dirección de *wallet* que se analice, el conjunto de métricas agregadas y el número de bloque hasta el que se ha procesado. La implementación de este contrato inteligente como repositorio de datos mejora sig-

nificativamente la eficiencia en análisis siguientes de una misma *wallet* pues en lugar de hacer de nuevo un barrido de todo el historial de bloques el sistema podrá recuperar directamente del contrato las métricas previamente calculadas y continuar el análisis únicamente a partir del último bloque registrado lo que reduce drásticamente la carga computacional y el tiempo de respuesta, especialmente para *wallets* con historiales extensos. Al registrar los datos agregados directamente en la *blockchain*, se crea una capa de persistencia independiente y robusta, lo cual garantiza que las métricas ya analizadas permanezcan públicamente accesibles y verificables de forma descentralizada, incluso en caso de una caída del servicio *off-chain*.

A continuación se detallan cada una de las métricas que se han considerado se deben extraer por el proveedor de datos, fundamentando su selección y explicando su relevancia para la caracterización del perfil de una *wallet*:

- Longevidad: Tiempo transcurrido desde la primera transacción que realizó la cuenta. Una mayor longevidad puede indicador una cuenta establecida y con una trayectoria más larga en la red, lo que entre otras cosas podría significar menos posibilidades de haber sido creada para fines maliciosos.

- Transacciones salientes: Corresponde al número total de transacciones iniciadas por la dirección de la *wallet*, donde esta actúa como remitente. Indica la proactividad de la cuenta en el inicio de operaciones.

- Transacciones entrantes: Es el número total de transacciones donde la dirección de la *wallet* figura como destinataria. Refleja la capacidad de la cuenta para recibir valor o interactuar con otras entidades.

- Transacciones fallidas: Cuantifica el número de transacciones iniciadas por la *wallet* que no se ejecutaron correctamente en la *blockchain* (ej, por falta de gas, errores de contrato). Una alta incidencia de transacciones fallidas podría sugerir inexperiencia o intentos de interacción con contratos inadecuados.

- Total de gas usado: Suma de todas las unidades de gas consumidas por las transacciones originadas o que involucraron a la *wallet*. El gas es una medida del esfuerzo computacional requerido, por lo que un alto consumo de gas indica una actividad intensiva o interacciones con contratos complejos.

- Tarifa promedio: Costo promedio pagado por transacción en términos de la moneda nativa de la *blockchain* (ej, ETH). Se calcula dividiendo el total de tarifas pagadas por el número de transacciones. Esta métrica puede reflejar la disposición de la cuenta a pagar por la prioridad de sus transacciones o su habilidad para optimizar los costos de gas.

- Contratos creados: Número de contratos inteligentes que han sido desplegados en la *blockchain* por la dirección de la *wallet*. La creación de contratos es una actividad significativa que a menudo se asocia con desarrolladores o entidades que buscan proveer nuevas funcionalidades o servicios en la red, indicando un rol activo en la expansión del ecosistema.

- Tokens ERC-20 [17] distintos: Número de contratos de *tokens* fungibles únicos (conformes al estándar ERC-20) con los que la *wallet* ha interactuado. Una mayor diversidad puede indicar una participación activa en el ecosistema DeFi o el uso de múltiples dApps que utilizan estos *tokens*.

- NFTs ERC-721 [18] distintos: Número de contratos de *tokens* no fungibles únicos (conformes al estándar ERC-721) con los que la *wallet* ha interactuado. Indica la implicación de la cuenta en el mercado de NFTs, ya sea como coleccionista, creador o comerciante, reflejando su participación en este segmento del ecosistema Web3.

- Total de días activos: Número de días únicos en los que la *wallet* ha registrado al menos una transacción (entrante o saliente). Esta métrica mide la regularidad y persistencia de la participación de la cuenta en la red a lo largo del tiempo, diferenciando entre actividad esporádica y un compromiso más sostenido.

La utilidad fundamental de este proveedor de datos de actividad *on-chain* reside en la adaptabilidad de sus métricas a diversos casos de uso. Si bien la presente tesis mostrará su aplicabilidad en la construcción de un sistema de reputación específico para un mercado P2P, es importante destacar que el conjunto de métricas propuesto anteriormente no está intrínsecamente ligado a una única interpretación o ponderación de reputación. Lo que queremos es ofrecer una capa base de información cuantitativa sobre el comportamiento de una *wallet*, que pueda ser consumida, adaptada e interpretada por diversas entidades o aplicaciones externas donde diferentes sistemas de reputación podrían asignar pesos variables a estas mismas métricas, o incluso descartar algunas, en función de sus contextos y objetivos particulares.

Por tanto se debe destacar que no se pretende proponer un modelo único de reputación, sino facilitar la creación de múltiples modelos al proveer, de manera accesible y estandarizada, las métricas *on-chain* que permitan a terceros diseñar e implementar sus sistemas de reputación alineados con sus necesidades específicas.

2.2. Sistema de reputación en mercado P2P.

Los mercados P2P, por su propio diseño inherente, está orientado a facilitar las interacciones directas entre individuos, eliminando o reduciendo significativamente la

necesidad de intermediarios centralizados. Aunque esto ofrece ventajas como menores costos, mayor eficiencia y resistencia a la censura, también introduce desafíos particulares en cuanto a la evaluación de la fiabilidad de las contrapartes. En ausencia de una entidad central que verifique identidades y modere disputas, los participantes se enfrentan a una mayor incertidumbre, lo que se acentúa en entornos basados en *blockchain*, ya que las interacciones a menudo ocurren entre direcciones seudónimas, lo que dificulta la atribución de un historial de comportamiento verificable por lo que es conveniente disponer de un mecanismo para cuantificar la confiabilidad.

2.2.1. Métricas fundamentales para el cálculo de la reputación en el mercado P2P.

Para la construcción de este ejemplo de sistema de reputación, seleccionamos cuatro métricas fundamentales, ya explicadas en la sección 2.1. Estas métricas, que en conjunto buscan ofrecer una perspectiva equilibrada del comportamiento de una cuenta, son: la Longevidad (L); el Volumen Total de Transacciones Exitosas (V), que contabiliza el número total de operaciones completadas correctamente (representa la suma de las transacciones correctas salientes y entrantes); las Transacciones Fallidas (F) y la Frecuencia de Actividad (FA) que evalúa la consistencia con la que una cuenta ha estado activa en relación con su tiempo total de existencia en la red. Estas métricas específicas se procesarán para generar una única puntuación de reputación.

2.2.2. Normalización y ponderación de las métricas.

Dada la heterogeneidad en las escalas y unidades de las métricas seleccionadas (días, número de transacciones, etc.), un paso esencial es su normalización a una escala común antes de cualquier combinación. Para este prototipo, se ha optado por una escala estandarizada de 0 a 5. En esta escala, una puntuación de 0 representa la contribución positiva más baja o el impacto negativo más alto (como en el caso de las transacciones fallidas), mientras que 5 denota la contribución positiva máxima.

La normalización se aborda de manera lineal definiendo umbrales para cada una de las métricas:

- Longevidad Normalizada (L_n), se considera un umbral de 730 días (aproximadamente dos años); alcanzar o superar este periodo otorga 5 puntos, escalándose linealmente para periodos menores ($L_n = \text{mín}(5, (\text{dias\_longevidad}/730) \cdot 5)$).
- Volumen de Transacciones Exitosas Normalizado (V_n) se establece con un umbral de 500 transacciones para obtener 5 puntos, con una escala lineal para volúmenes inferiores ($V_{\text{norm}} = \text{mín}(5, (\text{num\_tx\_exitosas}/500) \cdot 5)$).

- Transacciones Fallidas Normalizado (F_n): su impacto tiene un efecto inverso en la puntuación; se parte de 5 puntos para cero fallos, y cada cuatro transacciones fallidas restan un punto, hasta un mínimo de 0 puntos: ($F_{\text{norm}} = \text{máx}(0, 5 - (\text{num\_tx\_fallidas}/4))$).
- Frecuencia de Actividad Normalizada (FA_n): Esta métrica se calcula primero como una proporción: $F_R = (T_A/\,\text{máx}(1, D_{Tx}))$, donde T_A representa total de días activos y D_{TX} días desde la primera transacción. Luego se calcula $FA_n = \text{mín}(5, (F_R/0{,}50) \cdot 5)$ donde estar activo al menos 50 % de los días representa la puntuación máxima.

Estos umbrales (730 días, 500 transacciones, 20 transacciones fallidas equivalentes a 0 puntos, y una frecuencia de actividad relativa del 50 %) se han mantenido en cifras fáciles de conceptualizar para asegurar que la lógica de normalización sea lo más transparente posible, permitiendo que el usuario pueda intuir razonablemente cómo su actividad se traduce en su reputación final. Es importante destacar que además son valores solo para ilustrar el ejemplo, en un contexto real se refinarían mediante un análisis estadístico de la distribución de las métricas considerando cual es una población representativa de usuarios de la *blockchain* y buscando puntos de inflexión o valores que se correlacionen con diferentes niveles de actividad y fiabilidad.

2.2.3. Cálculo de la Puntuación de Reputación Final

Una vez que las cuatro métricas seleccionadas han sido normalizadas a la escala común de 0 a 5, la puntuación de reputación final (R) se determina mediante un promedio ponderado. Este enfoque permite asignar una importancia diferencial a cada métrica, adaptando el sistema a las prioridades del caso de uso particular. Para asegurar una interpretación directa de los pesos como contribuciones porcentuales a la puntuación total, se establece la restricción de que la suma de todos los pesos debe ser igual a 1 ($w_L + w_V + w_F + w_{FA} = 1$).

Con esta condición, la fórmula para calcular R es:

$$R = w_L \cdot L_n + w_V \cdot V_n + w_F \cdot F_n + w_{FA} \cdot FA_n$$

En esta ecuación, L_n, V_n, F_n, FA_n representan las puntuaciones normalizadas previamente calculadas para Longevidad, Volumen, Transacciones Fallidas y Frecuencia de Actividad, respectivamente. Los términos w_L, w_V, w_F, w_{FA} son los pesos asignados a cada una de estas métricas, cumpliendo que su suma es 1. Por ejemplo, teniendo un mercado P2P donde la consistencia y la fiabilidad operativa son primordiales se asignaran los siguientes pesos: $w_L = 0{,}25$ (Longevidad), $w_V = 0{,}20$ (Volumen), $w_F = 0{,}30$ (Impacto de Fallidas), y $w_{FA} = 0{,}25$ (Frecuencia de Actividad). Dado que $0{,}15 + 0{,}25 + 0{,}30 + 0{,}30 = 1{,}0$, estos pesos indican directamente la contribución de

cada métrica normalizada a la puntuación final. La puntuación de reputación resultante se encontrará también dentro de la escala de 0 a 5, facilitando su interpretación directa.

2.2.4. Interpretación y Aplicación de la Puntuación

Una puntuación de reputación comprendida entre 0 y 5 ofrece una forma intuitiva para que los usuarios de un mercado P2P evalúen a sus contrapartes. Por ejemplo, puntuaciones entre 4 y 5 podrían indicar una alta reputación, sugiriendo usuarios establecidos, activos, consistentes y con un historial operativo limpio. Valores entre 3 y 4 señalarían una buena reputación, quizás con un historial sólido pero menos extenso. Puntuaciones intermedias, entre 2 y 3, podrían corresponder a usuarios con actividad limitada o inconsistente, o con algunos errores, aconsejando mayor cautela. Finalmente, puntuaciones por debajo de 2 sugerirían una baja o muy baja reputación, asociadas a perfiles nuevos con escasa actividad, un número significativo de problemas, o un alto riesgo percibido. En la sección 4.2 se analizarán más a detalle estas interpretaciones.

Es fundamental reconocer que este sistema de reputación prototipo, aunque funcional y útil como demostración de concepto, presenta ciertas limitaciones inherentes a su diseño simplificado. La sensibilidad de la puntuación final a la elección de los umbrales de normalización y a los pesos asignados a cada métrica es una consideración importante, ya que estos parámetros son subjetivos y podrían requerir una calibración cuidadosa basada en datos empíricos y retroalimentación de la comunidad. Existe también la posibilidad de que actores intenten manipular el sistema inflando artificialmente ciertas métricas. Además, la relevancia y ponderación óptima de las métricas pueden variar según el contexto específico de la aplicación *blockchain*, y este modelo está diseñado con un mercado P2P generalista en mente.

Capítulo 3

Detalles de Implementación y Experimentos

El sistema se ha desarrollado en *Python* aprovechando un ecosistema de bibliotecas consolidadas para la interacción con *blockchains*, el desarrollo de servicios *web* y la creación de interfaces de usuario, además se utilizó *Solidity* [19] como lenguaje de programación para el contrato inteligente. A continuación, se describen las herramientas empleadas, la arquitectura general del sistema y la funcionalidad específica de cada uno de sus módulos.

3.1. Herramientas y Bibliotecas Utilizadas

La elección de las herramientas se basó en la robustez, la popularidad en la comunidad de desarrollo y la facilidad de integración. Las bibliotecas principales utilizadas son:

- **Streamlit [20]:** Es un *framework* de código abierto en *Python* que permite crear y compartir aplicaciones *web* para proyectos de forma rápida y sencilla. En este proyecto, se utiliza para construir la interfaz de usuario principal. Mediante esta interfaz el usuario puede configurar la conexión a la *blockchain* e indicar las *wallets* que quiere analizar.

- **Web3 [21]:** Es la biblioteca de *Python* de referencia para interactuar con nodos de *Ethereum*. Facilita la comunicación con la *blockchain* para realizar tareas como consultar el estado de la red (ej. número de bloque actual), leer datos de contratos inteligentes, enviar transacciones y firmarlas digitalmente. Es el pilar fundamental para la comunicación entre nuestra lógica de análisis y la *blockchain*.

- **FastAPI [22]:** Un moderno y rápido *framework web* para construir *APIs* con *Python*. Se eligió por su excelente rendimiento, su capacidad para generar documentación de *API* de forma automática y su sistema de validación de datos basado en *Pydantic*. *FastAPI* expone la lógica de análisis como un *endpoint RESTful*, permitiendo que otros servicios o aplicaciones consuman los datos de reputación de forma programática.
- **Uvicorn [23]:** Es un servidor *web ASGI (Asynchronous Server Gateway Interface)* ligero y rápido. Es el servidor recomendado para ejecutar aplicaciones *FastAPI* ya que gestiona las peticiones *HTTP* de manera asíncrona y eficiente.
- **python-dotenv:** Esta biblioteca se encarga de cargar variables de entorno desde un archivo `.env` en el directorio del proyecto. Se utiliza para gestionar de forma segura información sensible, como claves privadas y direcciones de contratos, sin necesidad de incluirlas directamente en el código fuente, lo cual es una práctica de seguridad fundamental.

3.2. Arquitectura y Componentes Funcionales

La implementación del sistema se organiza en torno a un conjunto de módulos donde cada uno es responsable de una funcionalidad específica.

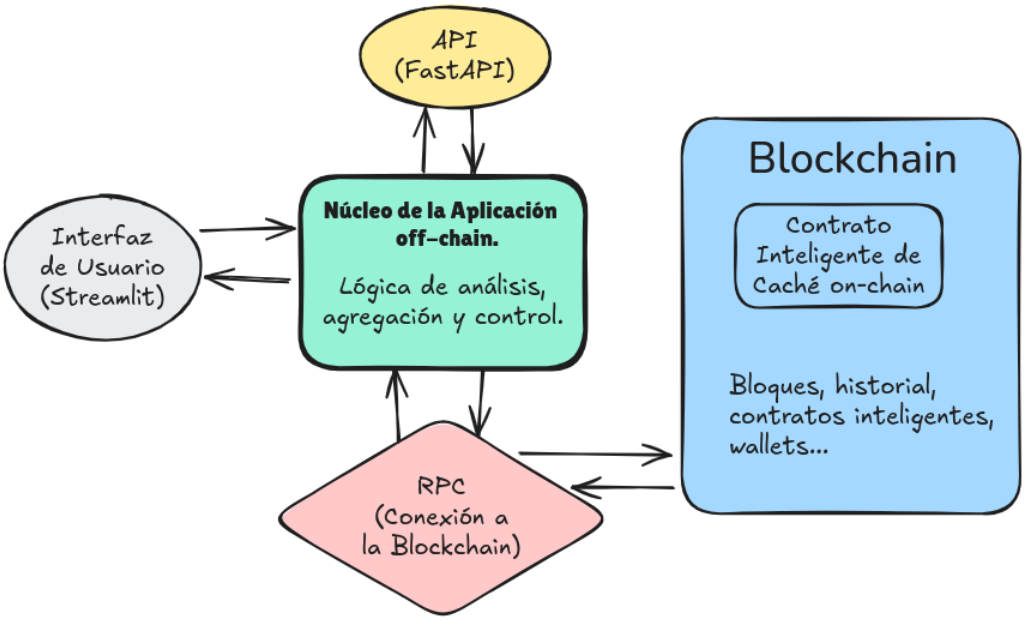

Figura 3.1: Arquitectura y flujo de trabajo.

El flujo de trabajo se inicia cuando un usuario, a través de la interfaz (*Streamlit*) o un sistema externo mediante la *API* (*FastAPI*), solicita el análisis de una *wallet*. El núcleo de la aplicación *off-chain* gestiona la petición ejecutando un proceso optimizado en tres pasos clave:

1. Realiza una llamada de solo lectura (*call*) al contrato inteligente para obtener los datos previamente cacheados y el último bloque analizado.

2. Procesa únicamente los nuevos bloques de la *blockchain* obtenidos a través del nodo *RPC*, evitando re-analizar el historial completo.

3. Envía una transacción (*transaction*) para actualizar el caché *on-chain* con las métricas consolidadas. Finalmente, el resultado completo y actualizado del análisis se devuelve en formato *JSON* al cliente que originó la consulta.

El código `app.py` actúa como el punto de entrada y orquestador del sistema. Implementa una arquitectura híbrida única que al ejecutarse, inicia una aplicación *web* interactiva y, simultáneamente, lanza un servidor de *API RESTful* (*FastAPI*) en un hilo (*thread*) de ejecución separado. Este diseño permite ofrecer, desde un único proceso, tanto una interfaz gráfica para el usuario final como un servicio programático para sistemas externos. La comunicación entre la aplicación principal y la *API* se logra mediante un diccionario de estado compartido (`SHARED_STATE`), que contiene las instancias activas de conexión a la *blockchain* permitiendo que la *API* acceda y reutilice la misma conexión configurada desde la interfaz gráfica sin necesidad de reinicializarla.

```
def run_api():
    """Ejecuta el servidor Uvicorn en un hilo."""
    uvicorn.run(api_app, host="0.0.0.0", port=8000, log_level="info")

def main():
    """Funci n principal de la aplicaci n Streamlit."""
    st.set_page_config(page_title="Sistema de Reputaci n", layout="wide")
    st.title("Proveedor de datos de actividad on-chain")

    # instancia de LocalStorage, para guardar el estado de la aplicaci n
    localS = LocalStorage()

    if not CONTRACT_ABI:
        st.error(f"Error cr tico: No se pudo cargar el ABI del contrato desde {CONTRACT_JSON_PATH}.")
        st.stop()
```

```
16
17      # inicia el hilo de la API si no se ha iniciado a n
18      if 'api_thread_started' not in st.session_state:
19          st.session_state.api_thread_started = True
20          # se usa un 'daemon thread' para que se cierre
    autom ticamente cuando la app principal se detenga
21          api_thread = threading.Thread(target=run_api, daemon=True)
22          api_thread.start()
23          st.toast("Servicio de API iniciado en http://localhost:8000"
    , icon="    ")
24
25      if 'initialized' not in st.session_state:
26          st.session_state.initialized = True
27          st.session_state.w3 = None
28          st.session_state.contract = None
29
30          st.session_state.app_config = localS.getItem("app_config")
    or {}
31          st.session_state.analysis_result = localS.getItem("
    analysis_result") or {}
32
33          config = st.session_state.app_config
34          if config.get("rpc_url") and config.get("contract_address"):
35              w3 = blockchain_utils.connect_to_node(config["rpc_url"],
     config["port"])
36              if w3:
37                  contract = blockchain_utils.get_contract_instance(w3
    , config["contract_address"])
38                  if contract:
39                      st.session_state.w3 = w3
40                      st.session_state.contract = contract
41
42                      # compartir estado con la API
43                      SHARED_STATE["w3"] = w3
44                      SHARED_STATE["contract"] = contract
45
46                      st.toast(f"  Reconectado   autom ticamente! API
    lista.", icon="    ")
47
48      tab_config, tab_analysis = st.tabs(["1. Configuraci n", "2.
    An lisis de Wallet"])
49
50      with tab_config:
51          ui_components.display_config_tab(localS)
52
53      with tab_analysis:
54          ui_components.display_analysis_tab(localS)
```

Ejemplo de código 3.1: Código de app.py, representa el punto de entrada al sistema.

3.2.1. Puntos de Entrada: Interfaz de Usuario y API

El sistema ofrece dos puntos de entrada para iniciar un análisis, atendiendo tanto a usuarios finales como a integraciones programáticas. La capa de presentación se construye con *Streamlit* y su lógica está modularizada para mantener la claridad del código. Este componente renderiza una interfaz de usuario organizada en dos pestañas. La primera (Çonfiguración") permite al usuario introducir los parámetros de conexión (*RPC*, dirección del contrato) y establecer la conexión. La segunda (."Análisis de Wallet") proporciona un campo para introducir la dirección a analizar y un botón para iniciar el proceso, mostrando posteriormente los resultados obtenidos en un formato legible (*JSON*). El uso de `streamlit-local-storage` mejora la usabilidad al persistir la configuración y los últimos resultados en el navegador del usuario.

Figura 3.2: Interfaz de usuario desarrollada con Streamlit.

Para permitir la integración con otros sistemas, el código `src/api.py` define una *API RESTful* utilizando el *framework FastAPI* que expone un único *endpoint* princi-

pal que acepta peticiones *POST* con la dirección de una *wallet*. El módulo utiliza modelos de *Pydantic* (`WalletRequest`, `WalletResponse`) para validar automáticamente los datos de entrada y para estructurar las respuestas de salida, garantizando una *API* robusta y autodocumentada. Este *endpoint* invoca la misma lógica de análisis central que la interfaz de usuario, asegurando la consistencia de los datos independientemente de cómo se acceda al sistema.

```
class WalletRequest(BaseModel):
    """El JSON que el cliente debe enviar en su petici n."""
    wallet_address: str = Field(...,
                                example="0
   x7DF8Efa6D6f1CB5C4f36315e0ACb82B02Ae8B240",
                                description="Direcci n de la wallet
    a analizar en formato checksum o no.")

class ReputationMetrics(BaseModel):
    """El sub-modelo para las m tricas de reputaci n."""
    txIn: int
    txOut: int
    totalTxs: int
    failedTxs: int
    gasUsed: int
    feePaid: int
    contractsCreatedCount: int
    distinctErc20Count: int
    distinctNftCount: int
    activeDaysCount: int
    firstTxTimestamp: int

class WalletResponse(BaseModel):
    """El JSON que la API devolver  en una respuesta exitosa."""
    wallet_address: str = Field(..., description="La direcci n
   analizada en formato checksum.")
    last_block_analyzed: int = Field(..., description="El  ltimo
   n mero de bloque que se consider  en el an lisis.")
    metrics: ReputationMetrics
    status: str = "success"
    message: str = "Reputation data retrieved successfully."


# instancia de FastAPI
api_app = FastAPI(
    title="API de Reputaci n de Wallets",
    description="Una API para obtener m tricas de reputaci n on-
   chain.",
    version="1.0.0"
)

```

```
37 # estado compartido para la conexi n a la blockchain
38 SHARED_STATE = {
39     "w3": None,
40     "contract": None
41 }
42
43 def get_shared_state():
44     """Dependencia de FastAPI para obtener el estado compartido."""
45     if not SHARED_STATE.get("w3") or not SHARED_STATE.get("contract"
       ):
46         raise HTTPException(
47             status_code=503,
48             detail="Servicio no disponible. La aplicaci n principal
        a n no est  conectada a la blockchain."
49         )
50     return SHARED_STATE
51
52
53 # !--- Endpoints de la API ---
54
55 @api_app.get("/", tags=["Status"])
56 def read_root():
57     """Endpoint de estado para verificar que la API est
       funcionando."""
58     return {"status": "API de Reputaci n de Wallets est  en l nea
       ."}
59
60
61 @api_app.post("/analyze", response_model=WalletResponse, tags=["
       An lisis"])
62 def analyze_wallet(
63     request: WalletRequest,
64     state: dict = Depends(get_shared_state)
65 ):
66     """
67     Analiza una wallet y actualiza los datos en el smart contract.
68     """
69     w3 = state["w3"]
70     contract = state["contract"]
71
72     if not Web3.is_address(request.wallet_address):
73         raise HTTPException(status_code=400, detail="La direcci n
       de la wallet proporcionada no es v lida.")
74
75     checksum_address = w3.to_checksum_address(request.wallet_address
       )
76
77     try:
```

```
78          # Preparamos las credenciales del owner
79          owner_address = None
80          owner_pk = None
81
82          if OWNER_ADDRESS_ENV and OWNER_PRIVATE_KEY:
83              owner_address = OWNER_ADDRESS_ENV
84              owner_pk = OWNER_PRIVATE_KEY
85          else:
86              raise HTTPException(
87                  status_code=503,
88                  detail="Credenciales del owner no configuradas. No
    se puede actualizar el contrato."
89              )
90
91          final_metrics, end_block = analysis.
    run_full_analysis_and_update(
92              w3, contract, checksum_address, owner_address, owner_pk
93          )
94
95          # Formatear y devolver la respuesta
96          final_metrics["feePaid"] = str(final_metrics["feePaid"])
97          final_metrics["gasUsed"] = str(final_metrics["gasUsed"])
98
99          return WalletResponse(
100             wallet_address=checksum_address,
101             last_block_analyzed=end_block,
102             metrics=ReputationMetrics(**final_metrics),
103             message=f"Reputation data retrieved. On-chain update was
     {'attempted' if owner_pk else 'skipped'}."
104         )
105
106     except Exception as e:
107         raise HTTPException(status_code=500, detail=f"Error interno
    del servidor durante el an lisis: {str(e)}")
```

Ejemplo de código 3.2: Definición del endpoint de la API con FastAPI.

3.2.2. Configuración y conexión con la Blockchain

Una vez que se inicia una solicitud de análisis, el primer paso es establecer una conexión segura y configurada con la *blockchain* siendo una funcionalidad esencial para la operatividad del sistema. El módulo `config.py` centraliza la gestión de parámetros críticos, como la carga del *ABI* (metadatos) del contrato desde su archivo *JSON* y la obtención de datos sensibles (direcciones y claves privadas para interactuar con la *Blockchain*) desde un archivo `.env`. Por su parte, `blockchain_utils.py` abstrae la lógica de conexión, proporcionando funciones para establecer la comunicación con un

nodo *RPC* y para instanciar el objeto del contrato inteligente, que son fundamentales para cualquier operación posterior.

```
    # carga las variables de entorno desde el archivo .env
    load_dotenv()

    # ! --- Constantes del proyecto ---
    # Orden de las m tricas , debe coincidir con el contrato y el an lisis
    METRIC_KEYS_ORDER = [
        "txIn", "txOut", "totalTxs", "failedTxs", "gasUsed", "feePaid",
        "contractsCreatedCount", "distinctErc20Count", "distinctNftCount",
        "activeDaysCount", "firstTxTimestamp"
    ]

    # ruta base del proyecto para construir rutas absolutas
    BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
    CONTRACT_JSON_PATH = os.path.join(BASE_DIR, 'contracts', 'walletDataCache.sol', 'WalletDataCache.json')

    # ! --- Variables de Entorno ---
    OWNER_ADDRESS_ENV = os.getenv("OWNER_ADDRESS", "")
    OWNER_PRIVATE_KEY = os.getenv("OWNER_PRIVATE_KEY")
    CONTRACT_ADDRESS_ENV = os.getenv("CONTRACT_ADDRESS", "")


    def load_contract_abi():
        """Carga el ABI del contrato desde el archivo JSON."""
        try:
            with open(CONTRACT_JSON_PATH) as f:
                contract_json = json.load(f)
                return contract_json['abi']
        except FileNotFoundError:
            return None
        except (json.JSONDecodeError, KeyError):
            return None

    CONTRACT_ABI = load_contract_abi()
```

Ejemplo de código 3.3: Código de configuración para la interacción con la blockchain.

3.2.3. Núcleo de Análisis y Optimización *On-Chain*

Con una conexión establecida, el núcleo computacional del sistema que procesa los datos *on-chain* reside en `src/analysis.py`. Su función principal, `process_blocks`, itera sobre el rango de bloques indicado examinando cada transacción y sus recibos para extraer las métricas fundamentales. Además, inspecciona los *logs* de eventos *Transfer* para identificar y contar las interacciones con *tokens ERC-20* y *NFTs*. Una función de búsqueda, `get_first_tx_timestamp`, optimiza la obtención de la fecha de la primera actividad de la *wallet*. Una característica clave del sistema es el uso de un contrato inteligente como capa de caché para optimizar los análisis. La interacción, facilitada por `src/blockchain_utils.py`, sigue un flujo optimizado:

- **Lectura del caché:** Antes de cada análisis, la función `get_cached_data_from_contract` realiza una llamada de solo lectura al contrato para recuperar las métricas ya calculadas y el último bloque analizado.
- **Procesamiento de nuevos datos:** El sistema procesa únicamente los bloques nuevos que se han generado desde el último análisis.
- **Actualización del caché:** Tras procesar los bloques nuevos, la función `update_data_in_contract` construye, firma con la clave privada del propietario y envía una transacción para actualizar el estado del contrato con los datos consolidados.

Este mecanismo evita re-analizar la historia completa de la *blockchain* en cada consulta, reduciendo drásticamente el tiempo de cálculo y los costos computacionales.

```
1  def process_blocks(w3: Web3, address: str, start_block: int,
       end_block: int):
2      """Procesa un rango de bloques para extraer m tricas de
       reputaci n."""
3
4      if start_block > end_block:
5          return None
6
7      address = w3.to_checksum_address(address)
8
9      stats = {key: 0 for key in METRIC_KEYS_ORDER}
10     stats_sets = {
11         "contracts_created": set(), "seen_erc20": set(),
12         "seen_nfts": set(), "active_days": set()
13     }
14
15     TRANSFER_SIG = "0x"+ w3.keccak(text="Transfer(address,address,
       uint256)").hex()
```

```
    ERC165_SIG = w3.keccak(text="supportsInterface(bytes4)")[:4].hex()
    ERC721_INTERFACE_ID = "0x80ac58cd"

    for b in range(start_block, end_block + 1):
        try:
            block = w3.eth.get_block(b, full_transactions=True)
            time_block = datetime.datetime.fromtimestamp(block.timestamp).strftime("%Y-%m-%d")

            for tx in block.transactions:
                is_relevant = (tx.get('to') == address) or (tx.get('from') == address)
                if is_relevant:
                    stats["totalTxs"] += 1
                    stats_sets["active_days"].add(time_block)

                    receipt = w3.eth.get_transaction_receipt(tx.hash)
                    stats["gasUsed"] += receipt.gasUsed
                    stats["feePaid"] += receipt.gasUsed * tx.gasPrice

                    if tx['from'] == address:
                        stats["txOut"] += 1
                        if tx.get('to') is None:
                            stats_sets["contracts_created"].add(receipt.contractAddress)

                    if tx.get('to') == address:
                        stats["txIn"] += 1

                    if receipt.status == 0:
                        stats["failedTxs"] += 1

            logs = w3.eth.get_logs({"fromBlock": b, "toBlock": b, "topics": [TRANSFER_SIG]})
            for log in logs:
                log_address = w3.to_checksum_address(log['address'])
                try:
                    data = ERC165_SIG + ERC721_INTERFACE_ID[2:].rjust(64, '0')
                    res = w3.eth.call({"to": log_address, "data": data}, b)
                    if int(res.hex(), 16):
                        stats_sets["seen_nfts"].add(log_address)
                    else:
                        stats_sets["seen_erc20"].add(log_address)
```

```
                except Exception:
                    stats_sets["seen_erc20"].add(log_address)

        except Exception as e:
            print(f"No se pudo procesar el bloque {b}: {e}")
            continue

    stats["contractsCreatedCount"] = len(stats_sets["contracts_created"])
    stats["distinctErc20Count"] = len(stats_sets["seen_erc20"])
    stats["distinctNftCount"] = len(stats_sets["seen_nfts"])
    stats["activeDaysCount"] = len(stats_sets["active_days"])

    return stats
```

Ejemplo de código 3.4: Función principal de análisis de bloques.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "hardhat/console.sol";  // debug con console.log en Solidity

contract WalletDataCache {
    address public owner;

    struct WalletMetrics {
        uint256 txIn;
        uint256 txOut;
        uint256 totalTxs;
        uint256 failedTxs;
        uint256 gasUsed;
        uint256 feePaid; // En Wei
        uint256 contractsCreatedCount;
        uint256 distinctErc20Count;
        uint256 distinctNftCount;
        uint256 activeDaysCount;
        uint256 firstTxTimestamp;
    }

    mapping(address => WalletMetrics) public walletMetricsCache;
    mapping(address => uint256) public lastProcessedBlock;

    event WalletDataUpdated(address indexed wallet, uint256 lastBlock);
    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

    constructor() {
```

```
30            owner = msg.sender;
31            console.log("WalletDataCache deployed by:", msg.sender);
32            emit OwnershipTransferred(address(0), owner);
33        }
34
35        modifier onlyOwner() {
36            require(msg.sender == owner, "WalletDataCache: Caller is
     not the owner");
37            _;
38        }
39
40        function updateWalletData(
41            address _wallet,
42            WalletMetrics calldata _metrics,
43            uint256 _blockNumber
44        ) external onlyOwner {
45            walletMetricsCache[_wallet] = _metrics;
46            lastProcessedBlock[_wallet] = _blockNumber;
47            emit WalletDataUpdated(_wallet, _blockNumber);
48        }
49
50        function getWalletData(address _wallet)
51            external
52            view
53            returns (WalletMetrics memory, uint256)
54        {
55            return (walletMetricsCache[_wallet], lastProcessedBlock[
    _wallet]);
56        }
57
58        function transferOwnership(address newOwner) external
    onlyOwner {
59            require(newOwner != address(0), "WalletDataCache: New
    owner is the zero address");
60            owner = newOwner;
61            emit OwnershipTransferred(msg.sender, newOwner);
62        }
63    }
```

Ejemplo de código 3.5: Código del contrato inteligente para caché on-chain.

Es importante señalar que la implementación de un propietario único (*owner*) deposita toda la responsabilidad de la seguridad y la administración del contrato inteligente en una sola entidad. Esto significa que la custodia de la clave privada de la cuenta del *owner* se convierte en un eslabón crítico del sistema. Un manejo inadecuado, como almacenar la clave en un dispositivo no seguro o perder las credenciales de recuperación, podría llevar a un bloqueo permanente del contrato o a su control por parte de un atacante. Por lo tanto, el usuario que asuma este rol debe tener un

conocimiento avanzado sobre prácticas de seguridad para garantizar la continuidad del servicio.

3.3. Experimentos

La experimentación se llevó a cabo en un entorno local para garantizar la reproducibilidad y eliminar las variables de latencia de una red pública. Se utilizó *Ganache* [24], una *blockchain* personal para el desarrollo de *Ethereum* que se ejecuta localmente, permitiendo simular interacciones sin incurrir en costos reales ni depender de servicios externos. Se configuró una instancia de *Ganache* con 100 cuentas (*wallets*) pre-financiadas con *ETH* de prueba y con una creación de bloques cada 8 segundos.

Para generar un historial de transacciones, se ejecutó un *script* de simulación en *Python* (utilizando la biblioteca *Web3.py*) que generó aleatoriamente interacciones entre las 100 *wallets*, posteriormente se conectó la aplicación al nodo *RPC* de *Ganache* y se analizaron algunas direcciones tanto por la interfaz gráfica como por la *API* obteniendo un conjunto completo de métricas que reflejaban la actividad que se simuló. Ejemplo en *JSON* de métricas obtenidas para una *Wallet*:

```
{
    "txIn": 183,
    "txOut": 199,
    "totalTxs": 382,
    "failedTxs": 0,
    "gasUsed": 8022000,
    "feePaid": 42629468805000,
    "contractsCreatedCount": 0,
    "distinctErc20Count": 0,
    "distinctNftCount": 0,
    "activeDaysCount": 2,
    "firstTxTimestamp": 1749503551
}
```

Ejemplo de código 3.6: Ejemplo de salida de métricas en formato JSON.

Se comprobó la veracidad de los resultados obtenidos siendo los mismos que brinda la interfaz de *Ganache*, confirmando que el motor de análisis es capaz de diferenciar entre distintos tipos de comportamiento *on-chain* y cuantificarlos correctamente según las métricas definidas. Para medir la eficacia del mecanismo de caché, se realizó un experimento en dos fases para una *wallet* particularmente activa:

- **Sin caché:** Se ejecutó el análisis por primera vez. El sistema tuvo que procesar todos los bloques desde el bloque génesis hasta el bloque actual.
- **Con caché:** Se permitió que la simulación continuara, añadiendo 100 bloques nuevos con más transacciones. Luego, se volvió a analizar la misma *wallet*. Esta

vez, el sistema leyó el estado del contrato inteligente y solo procesó los 100 bloques nuevos.

Los tiempos de ejecución, medidos localmente, se muestran en la Tabla 3.1.

Tabla 3.1: Comparación de Tiempos de Análisis con y sin Caché

Tipo de Análisis	**Bloques a Procesar**	**Tiempo de Ejecución (aprox.)**
Sin caché	4230 bloques en total	25.4 segundos
Con caché	Más 100 bloques nuevos	0.6 segundos

Los resultados demuestran de manera concluyente que el mecanismo de caché en el contrato inteligente es extremadamente eficaz, reduciendo el tiempo de análisis considerablemente. Esta optimización es crucial para la viabilidad de un sistema de reputación que opere en redes públicas que pueden tener grandes cantidades de bloques.

Capítulo 4

Resultados para Sistema de Reputación en mercado P2P

4.1. Enfoque Metodológico: Análisis de Casos Teóricos

Como caso de uso del proveedor de datos *on-chain* se presentó un sistema de reputación para un mercado *P2P*, sin embargo su evaluación presenta desafíos metodológicos significativos. Una simulación empírica exhaustiva requeriría modelar complejas interacciones humanas a lo largo del tiempo, incluyendo disputas, acuerdos subjetivos y comportamientos estratégicos que son difíciles de simular con código. En consecuencia, y para validar la lógica y la sensibilidad del modelo propuesto en un entorno controlado, se ha optado por un análisis de algunos casos "teóricos".

Este enfoque consiste en definir una serie de arquetipos de usuario, donde cada uno representa un perfil de comportamiento plausible dentro de un mercado *P2P*. A cada arquetipo se le asignan valores de métricas crudas que reflejan su historial. Posteriormente, se aplica el algoritmo de cálculo de reputación (normalización y promedio ponderado, como se describe en la sección 2.2) para obtener una puntuación final. El objetivo es demostrar que el sistema es capaz de:

- Diferenciar cuantitativamente entre perfiles de usuario distintos.
- Asignar puntuaciones altas a comportamientos deseables (consistencia, fiabilidad, experiencia).
- Penalizar de manera efectiva los comportamientos negativos (transacciones fallidas).

- Producir resultados intuitivos y lógicos que se corresponden con la percepción humana de la confianza.

Para los siguientes cálculos, se utilizarán los pesos definidos en la sección 2.2.3: $w_L = 0{,}25$, $w_V = 0{,}20$, $w_F = 0{,}30$, y $w_{FA} = 0{,}25$. La asignación de estos pesos, aunque inherentemente subjetiva y adaptable a diferentes contextos de mercado, ha sido diseñada para reflejar un modelo de confianza que valora primordialmente la fiabilidad operativa y la consistencia del comportamiento. Se otorga el mayor peso a las Transacciones Fallidas ($w_F = 0{,}30$), ya que un alto número de fallos es el indicador más directo de riesgo o incompetencia técnica, erosionando la confianza de manera significativa. La Longevidad ($w_L = 0{,}25$) y la Frecuencia de Actividad ($w_{FA} = 0{,}25$) reciben un peso considerable, pues juntas sugieren un compromiso sostenido y una presencia estable en el ecosistema. Finalmente, el Volumen ($w_V = 0{,}20$) tiene el peso más bajo, reconociendo que, si bien la actividad es importante, la calidad y la fiabilidad de las interacciones son más cruciales para construir una reputación sólida que el simple volumen transaccional.

4.2. Definición de Arquetipos y Resultados

Se han definido cinco arquetipos para cubrir un espectro representativo de usuarios:

- **Veterano Fiable:** Un usuario con un largo y extenso historial de transacciones exitosas, muy activo y con un número mínimo de fallos.
- **Usuario Nuevo y Prometedor:** Un participante reciente que, en poco tiempo, ha demostrado ser muy activo, consistente y fiable.
- **Participante Ocasional:** Un usuario que lleva tiempo en la red pero interactúa de forma esporádica, aunque de manera fiable cuando lo hace.
- **Operador de Riesgo:** Un usuario con un volumen de transacciones moderado pero con un historial notable de fallos, indicando poca fiabilidad o cuidado.
- **Cuenta Durmiente:** Una cuenta muy antigua pero con una actividad casi nula, representando una incógnita.

La Tabla 4.1 muestra los resultados del cálculo de la reputación para cada uno de estos arquetipos.

Tabla 4.1: Cálculo de la Puntuación de Reputación para Arquetipos de Usuario. Métricas Crudas

Métricas Crudas	**Veterano Fiable**	**Nuevo y prometedor**	**Participante Ocasional**	**Operador de Riesgo**	**Cuenta Durmiente**
Longevidad(días)	800	60	400	180	1000
Transacciones Éxitosas	650	80	30	120	5
Transacciones Fallidas	1	0	0	9	0
Días Activos	450	45	20	60	4

Tabla 4.2: Cálculo de la Puntuación de Reputación para Arquetipos de Usuario. Métricas Normalizadas

Métricas Normalizadas	**Veterano Fiable**	**Nuevo y prometedor**	**Participante Ocasional**	**Operador de Riesgo**	**Cuenta Durmiente**
Longevidad(L_n)	5	0.41	2.74	1.23	5
Volumen (V_n)	5	0.8	0.3	1.2	0.05
Transacciones Fallidas (F_n)	4.75	5	5	2.75	5
Frecuencia(FA_n)	5	5	0.5	3.33	0.04
Puntuación de Final	**4.92**	**3.01**	**2.37**	**2.21**	**2.77**

4.3. Discusión e Interpretación de los Resultados

El análisis de la Tabla 4.2 permite extraer conclusiones clave sobre el comportamiento del sistema de reputación:

- **Veterano Fiable (R = 4.92):** Como era de esperar, este perfil obtiene una puntuación casi perfecta. El sistema recompensa al máximo su longevidad, volumen, fiabilidad (una penalización mínima por un solo fallo) y su alta frecuencia de actividad. Este resultado valida que el modelo identifica y valora positivamente a los actores ideales de un mercado.

- **Usuario Nuevo y Prometedor (R = 3.01):** Esta puntuación intermedia-alta es muy ilustrativa. A pesar de su baja longevidad ($L_n = 0{,}41$), el sistema reconoce su impecable fiabilidad ($F_n = 5{,}00$) y su alta frecuencia de actividad ($FA_n = 5{,}00$).

- **Participante Ocasional (R = 2.37):** Este arquetipo recibe una puntuación

en el rango bajo-intermedio. Su fiabilidad es perfecta ($F_n = 5{,}00$), pero su bajo volumen y, sobre todo, su bajísima frecuencia de actividad ($V_n = 0{,}30$, $FA_n = 0{,}50$) impactan negativamente en la puntuación final. El sistema lo interpreta correctamente: es un usuario fiable cuando opera, pero su escasa participación lo convierte en un miembro menos establecido de la comunidad.

- **Operador de Riesgo (R = 2.21):** Este es un caso crítico para la validación. Aunque su volumen de transacciones (120) es superior al del nuevo usuario y al del participante ocasional, su puntuación final es la más baja entre los usuarios activos. Esto se debe al fuerte impacto negativo de sus 9 transacciones fallidas, que reducen su F_n a solo 2.75. Dado que F_n tiene un peso del 30 %, esta penalización es sustancial y demuestra que el sistema cumple su objetivo primordial: priorizar la fiabilidad operativa por encima del mero volumen de actividad.

- **Cuenta Durmiente (R = 2.77):** Esta cuenta recibe una puntuación baja, a pesar de su máxima longevidad y su historial limpio de fallos. El resultado es lógico: la casi inexistente actividad (V_n y FA_n cercanos a cero) la convierte en una entidad desconocida. El sistema la califica como de baja reputación no porque sea maliciosa, sino porque no existe un historial de comportamiento relevante sobre el cual construir confianza.

Para facilitar su aplicación práctica, se puede establecer un umbral clave: una puntuación por debajo de 3.0 debe ser considerada una señal de alerta que amerita precaución. Este umbral no distingue necesariamente a un actor malicioso de uno simplemente inexperto o inactivo, pero agrupa a todos los perfiles que carecen de un historial robusto y positivo. Por lo tanto, representa la línea divisoria entre los participantes establecidos y confiables y aquellos cuyo historial es insuficiente o problemático. El análisis de casos teóricos ha mostrado que el modelo de reputación propuesto es funcional, sensible y se alinea con las expectativas lógicas de un sistema de confianza. Es capaz de distinguir matices entre diferentes perfiles de usuario, penalizar eficazmente la falta de fiabilidad y valorar la consistencia, proporcionando una base cuantitativa sólida para apoyar la toma de decisiones.

Si bien el mercado *P2P* es un excelente caso de uso, este mecanismo de reputación basado en umbrales es directamente transferible a otros ecosistemas donde la confianza verificable es fundamental, como en un entorno académico descentralizado. Por ejemplo, un ecosistema digital universitario gestionado mediante una *DAO (Organización Autónoma Descentralizada)*, donde estudiantes y académicos interactúan *on-chain*. En este escenario, la reputación de un participante podría influir en el acceso a recursos y su capacidad de colaboración. Actividades como la participación en proyectos de investigación, el acceso a servidores de computación de alto rendimiento,

la solicitud de material de laboratorio, o incluso el peso del voto en decisiones de la comunidad, podrían estar condicionadas por su puntuación de reputación.

Este escenario académico demuestra la flexibilidad del sistema propuesto. El Proveedor de Datos de Actividad *on-chain* actúa como la capa de inteligencia fundamental que capitaliza la información objetiva de la *blockchain*, transformando datos abstractos de transacciones en una visión clara y accionable. Esta visión permite construir modelos de confianza que son funcionales y relevantes tanto en un mercado comercial *P2P* como en un entorno colaborativo y de alto valor.

Conclusiones

El presente trabajo de diploma abordó el desafío de cuantificar la confianza en ecosistemas descentralizados mediante el diseño e implementación de un sistema de reputación basado exclusivamente en datos de actividad *on-chain*. A través de una metodología que combinó la definición teórica, el desarrollo de un prototipo funcional y la experimentación controlada, se considera que se han alcanzado los objetivos fundamentales, obteniendo hallazgos significativos.

El principal logro ha sido la creación de un sistema de dos capas: un proveedor de datos de actividad y un modelo de reputación prototipo. El proveedor de datos demostró ser capaz de procesar el historial transaccional de una *wallet* en una *blockchain* compatible con *EVM*, extrayendo un conjunto de métricas cuantitativas que reflejan su comportamiento. La implementación de un contrato inteligente como mecanismo de persistencia y *caché on-chain* fue clave en la optimización del sistema. Los experimentos realizados confirmaron de manera concluyente la eficacia de este enfoque, logrando una reducción drástica en los tiempos de análisis para consultas subsecuentes y mejorando significativamente el rendimiento.

El sistema de reputación, desarrollado como caso de uso para un mercado *P2P*, mostró la utilidad de los datos extraídos. Mediante un proceso transparente de normalización y ponderación de métricas, el modelo fue capaz de generar una puntuación de reputación final intuitiva. El análisis de arquetipos teóricos demostró que el sistema puede diferenciar cuantitativamente entre perfiles de usuario diversos, asignando puntuaciones que se alinean con la percepción humana de la confianza y penalizando de forma efectiva comportamientos negativos como una alta tasa de transacciones fallidas.

A pesar de los resultados positivos, es importante reconocer las limitaciones inherentes al sistema. En primer lugar, la escalabilidad y el rendimiento para el análisis inicial de *wallets* con un historial muy extenso siguen siendo un desafío y aunque el *caché* optimiza las consultas repetidas, la primera ejecución puede requerir recorrer una gran cantidad de bloques, un proceso que puede ser lento y computacionalmente intensivo, especialmente en redes públicas con millones de bloques. En segundo lugar, el rendimiento está intrínsecamente ligado a la latencia de la conexión con el nodo *RPC*, un nodo remoto o congestionado puede convertirse en un cuello de bo-

tella significativo. Finalmente, persiste una subjetividad evidente en la construcción de diferentes modelos de reputación, donde la selección de métricas y la asignación de sus pesos son decisiones de diseño complejas que merecen un profundo análisis del caso de uso, así como una continua revisión y retroalimentación de la opinión de sus usuarios. La justicia de una puntuación de reputación sigue siendo un debate abierto, y cualquier modelo numérico es una simplificación de la compleja realidad del comportamiento humano.

Recomendaciones

A partir de los hallazgos y limitaciones identificados en este trabajo, se proponen las siguientes líneas de continuidad y mejora para futuras investigaciones y desarrollos:

- **Refinamiento y expansión de las métricas:** El modelo actual se basa en un conjunto fundamental de métricas. Una futura línea de trabajo podría explorar la incorporación de indicadores más sofisticados, como el análisis del valor monetario de las transacciones, el análisis de grafos para medir la centralidad de una *wallet* en el ecosistema, o la reputación de las contrapartes con las que transacciona. La calibración de los umbrales de normalización y los pesos mediante análisis estadístico sobre grandes conjuntos de datos reales permitiría crear un modelo más robusto y adaptado a la dinámica de la red.

- **Optimización de la obtención de datos:** Para superar las limitaciones de rendimiento en las *blockchains* públicas, se recomienda explorar la integración con servicios de indexación de terceros, como *The Graph*, *Dune Analytics*, o las *APIs* avanzadas de proveedores como *Alchemy* o *Infura*. Algunas de estas plataformas ofrecen datos pre-procesados y optimizados para consultas, sin embargo, el uso de su pleno potencial generalmente requiere suscripciones de pago, cuya implementación excedía el alcance del presente trabajo.

- **Extensión a otras blockchains:** El sistema actual está diseñado para *blockchains* compatibles con *EVM*. Un paso lógico para ampliar su impacto sería desarrollar módulos de extracción de datos específicos para cada ecosistema convirtiendo la herramienta en una solución de reputación totalmente independiente a la cadena subyacente.

- **Investigación en resistencia a la manipulación:** Aunque el sistema actual se basa en datos objetivos, actores maliciosos podrían intentar manipular sus puntuaciones (por ejemplo, generando transacciones de bajo costo entre cuentas para inflar el volumen). Se recomienda investigar la implementación de mecanismos de defensa, como ponderar las métricas en función del costo de generarlas o integrar soluciones de identidad descentralizada para aumentar la resistencia a ataques *Sybil*.

- **Mejoras de seguridad y usabilidad en el Contrato Inteligente:** El modelo actual de propietario único pudiera introducir un punto central de fallo por lo que se recomienda investigar diseños más seguros y descentralizados como pudiera ser el uso de una cartera multifirma. Adicionalmente para asegurar el ciclo de vida y la capacidad de mejora del contrato se pudiera implementar un patrón de proxy actualizable permitiendo futuras modificaciones en la lógica sin perder el estado almacenado.

Bibliografía

[1] Yuval Noah Harari. *Nexus*. Trad. por Jordi Ros i Aragonès. Penguin Random House Grupo Editorial, 2024 (vid. pág. 1).

[2] Chrysanthos Dellarocas. «Analyzing the Economic Efficiency of eBay-like Online Reputation Reporting Mechanisms». En: *Proceedings of the ACM Conference on Electronic Commerce*. 2002. DOI: 10.2139/ssrn.289968 (vid. págs. 1, 4).

[3] Lingcong Wang, Kaiwen Tan y Yi Huang. «Reputation Analysis of E-commerce Products Based on Online Reviews—Take Amazon as an Example». En: *Journal of Economics and Public Finance* 6.2 (2020), págs. 128-145. DOI: 10.22158/jepf.v6n2p128 (vid. pág. 1).

[4] Tim Christiaens. «Trust and power in Airbnb's digital rating and reputation system». En: *Ethics and Information Technology* 27 (2025), págs. 1-13. DOI: 10.1007/s10676-025-09825-6 (vid. págs. 1, 5).

[5] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf. 2008 (vid. pág. 1).

[6] Vitalik Buterin. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf. 2014 (vid. pág. 1).

[7] Amazon. *Ayuda y Servicio de Atención al Cliente*. Consultado en 2024. La fuente original no especifica fecha de publicación (s.f.) (vid. pág. 5).

[8] A. Battah, Y. Iraqi y E. Damiani. «Blockchain-Based Reputation Systems: Implementation Challenges and Mitigation». En: *Electronics* 10.3 (2021), pág. 289. DOI: 10.3390/electronics10030289 (vid. págs. 6, 7, 10).

[9] Bruno Rodrigues y col. «On Trust, Blockchain, and Reputation Systems». En: ene. de 2022 (vid. págs. 6, 10).

[10] Junaid Arshad y col. «REPUTABLE - A Decentralized Reputation System for Blockchain-based Ecosystems». En: *IEEE Access* (jul. de 2022). DOI: 10.1109/ACCESS.2022.3194038 (vid. pág. 7).

[11] Audun Jøsang y Roslan Ismail. «The Beta Reputation System». En: *In: Proceedings of the 15th Bled Conference on Electronic Commerce* (ene. de 2002) (vid. pág. 7).

[12] Zhe Tu y col. «A Blockchain-based Trust and Reputation Model with Dynamic Evaluation Mechanism for IoT». En: *Computer Networks* 218 (oct. de 2022), pág. 109404. DOI: 10.1016/j.comnet.2022.109404 (vid. pág. 8).

[13] Francesco. *The Importance of On-Chain Reputation*. Substack. 2024. URL: https://fraxcesco.substack.com/p/the-importance-of-on-chain-reputation (vid. pág. 11).

[14] Gitcoin Team. *Everything You Need to Know About Gitcoin*. Gitcoin Blog. 2024. URL: https://www.gitcoin.co/blog/everything-you-need-to-know-about-gitcoin (vid. pág. 12).

[15] Octan Network. *Octan Reputation Analytics*. Documentation. 2024. URL: https://docs.octan.network/octan-docs-en/octan-reputation-analytics/ (vid. pág. 13).

[16] O. Aluko y A. Kolonin. «Proof-of-Reputation: An Alternative Consensus Mechanism for Blockchain Systems». En: *International Journal of Network Security & Its Applications (IJNSA)* 13.4 (2021). DOI: 10.5121/ijnsa.2021.13403 (vid. pág. 15).

[17] Fabian Vogelsteller y Vitalik Buterin. *EIP-20: Token Standard*. 2015. URL: https://eips.ethereum.org/EIPS/eip-20 (vid. pág. 20).

[18] William Entriken y col. *EIP-721: Non-Fungible Token Standard*. 2018. URL: https://eips.ethereum.org/EIPS/eip-721 (vid. pág. 20).

[19] Ethereum Team. *Solidity Language Documentation*. 2024. URL: https://soliditylang.org (vid. pág. 24).

[20] Streamlit Team. *Streamlit Documentation*. Snowflake Inc. 2024. URL: https://streamlit.io (vid. pág. 24).

[21] Ethereum Foundation. *Web3.py Documentation*. 2024. URL: https://web3py.readthedocs.io (vid. pág. 24).

[22] Sebastián Ramírez. *FastAPI Documentation*. 2024. URL: https://fastapi.tiangolo.com (vid. pág. 25).

[23] Tom Christie y collaborators. *Uvicorn: The lightning-fast ASGI server*. 2024. URL: https://www.uvicorn.org (vid. pág. 25).

[24] Truffle Suite. *Ganache: Personal Ethereum Blockchain*. ConsenSys. 2024. URL: https://trufflesuite.com/ganache/ (vid. pág. 37).