La investigación reportada en esta tesis es parte de los programas de investigación del CICESE (Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California).

La investigación fue financiada por el CONAHCYT (Consejo Nacional de Humanidades, Ciencias y Tecnologías).

Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias
en Electrónica y Telecomunicaciones
con orientación en Telecomunicaciones

Gestión de SLA (Service Level Agreement) aplicando Blockchain y Smart Contract para dominios de redes 6G

Tesis
para cubrir parcialmente los requisitos necesarios para obtener el grado de
Maestro en Ciencias

Presenta:

Ariel Irigoyen Castro

Ensenada, Baja California, México
2024

Tesis defendida por

Ariel Irigoyen Castro

y aprobada por el siguiente Comité

Dr. Jorge Enrique Preciado Velasco
Director de tesis

Dr. Miguel Ángel Alonso Arévalo

Dr. Jorge Torres Rodríguez

Dr. José Eleno Lozano Risk

Dra. María del Carmen Maya Sánchez
Coordinadora del Posgrado en Electrónica y Telecomunicacione

Dra. Ana Denise Re Araujo
Directora de Estudios de Posgrado

Resumen de la tesis que presenta **Ariel Irigoyen Castro** como requisito parcial para la obtención del grado de Maestro en Ciencias en Electrónica y Telecomunicaciones con orientación en Telecomunicaciones.

Gestión de SLA (Service Level Agreement) aplicando Blockchain y Smart Contract para dominios de redes 6G

Resumen aprobado por:

Dr. Jorge Enrique Preciado Velasco
Director de tesis

Mientras que en la red 4G todos los servicios son tratados por igual, en la red 5G se adopta una arquitectura basada en servicios los cuales se manejan de forma diferenciada. Para conformar esta arquitectura la tecnología *Network Slices* (NS) juega un rol fundamental ya que posibilita dividir la infraestructura física en varias redes lógicas las cuales están optimizadas para servicios y aplicaciones específicas. NS habilita un creciente mercado para interesados en arrendar segmentos de red 5G. El arrendamiento ocurre con el establecimiento de acuerdos de nivel de servicio (SLA) entre clientes y proveedores de NS, pero la gestión tradicional de los SLA presenta falta de transparencia para los clientes, gran volumen de trabajo para los proveedores, entorpecimiento del proceso de gestión por la intervención de actores externos en caso de disputas entre proveedores y clientes, y falta de automatización. Una forma de enfrentar estos problemas es aprovechar las bondades de las tecnologías *Smart Contract* (SC) y *Blockchain* (BC) como son la confianza, transparencia, seguridad, funcionamiento distribuido y automatización. Debido a lo anterior en este trabajo se diseña y valida un esquema de gestión de SLA de NS que utiliza la dupla BC y SC, de esta manera se proporciona un entorno transparente y confiable para la relación entre proveedores y clientes sin necesidad de intermediarios. Como parte de la validación del esquema se realizan pruebas de su flujo de ejecución en una BC local y en las *testnet Polygon Mumbai* y *Ethereum Sepolia*. Las pruebas realizadas en *testnet* estudian el desempeño en cuanto a costos y tiempo de ejecución de las funciones SC implementados como parte de esquema haciendo una comparación de los resultados para cada *testnet* demostrando el impacto en el rendimiento del esquema según la elección de la BC. Esta propuesta mejora la eficiencia y seguridad en la gestión de calidad de servicio para redes futuras, facilitando nuevos modelos de negocio y casos de uso exigentes, como los de redes B5G.

Palabras clave: 5G, Network Slice, SLA, SC, BC.

Abstract of the thesis presented **by Ariel Irigoyen Castro** as a partial requirement to obtain the Master of Science degree in Electronics and Telecommunication with orientation in Telecommunications

SLA (Service Level Agreement) Management applying Blockchain and Smart contract for 6G network domains

Abstract approved by:

Dr. Jorge Enrique Preciado Velasco
Thesis Director

While all services are treated equally in the 4G network, the 5G network handles services differently according to a service-based architecture. Network Slices (NS) technology plays a fundamental role in forming this architecture as it divides the physical infrastructure into several logical networks optimized for specific services and applications. NS enables a growing market for those leasing 5G network segments. Leasing occurs when customers and NS providers establish Service Level Agreements (SLAs). However, traditional SLA management presents a need for more transparency for customers, a significant workload for providers, a hindrance to the management process due to the intervention of external actors in case of disputes between providers and customers, and a lack of automation. One way to address these problems is to take advantage of the benefits of Smart Contract (SC) and Blockchain (BC) technologies, such as trust, transparency, security, distributed operation, and automation. Due to the above, this work designs and validates an NS SLA management scheme that uses the BC and SC pair, thus providing a transparent and reliable environment for the relationship between providers and customers without intermediaries. As part of the scheme validation, we tested its execution flow on a local BC and the Polygon Mumbai and Ethereum Sepolia testnets. The tests carried out on testnets study the performance in terms of costs and execution time of the SC functions implemented as part of the scheme, comparing the results for each testnet, demonstrating the impact on the scheme's performance according to the choice of BC. This proposal improves efficiency and security in quality of service management for future networks, facilitating new business models and demanding use cases, such as those for B5G networks.

Keywords: 5G, Network Slice, SLA, SC, BC.

Dedicatoria

A mi madre quien me dio la vida y me apoyó para ser mejor cada día.

A mi padre, por empujarme a pensar antes de cometer errores.

Agradecimientos

A mi director de tesis, el Dr. Jorge Preciado, por su especial dedicación, consejos, paciencia y apoyo incondicional.

A los miembros del comité de tesis Miguel Alonso, Jorge Torres y José Lozano por su generosa orientación, certeras correcciones y opiniones interesantes sobre el trabajo de investigación.

A mi amigo Lázaro, hermano y consejero, aquí en un país extranjero lejos de la familia.

A mi novia que ha sido luz y alegría en momentos de tensión durante el transcurso del posgrado.

A los profesores del CICESE por su dedicación y esfuerzo en la transmisión de conocimientos.

A todos los estudiantes y personal del Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California (CICESE) por su espíritu comunitario y de apoyo.

Al Consejo Nacional de Humanidades, Ciencia y Tecnología (CONAHCyT), por proporcionarme el apoyo económico para realizar mis estudios de maestría. No. CVU: 1247230

A México, por ser un país generoso y darme la grandiosa oportunidad de superarme profesionalmente.

Tabla de contenido

Resumen en español ii

Resumen en inglés iii

Dedicatoria iv

Agradecimientos v

Lista de Figuras ix

Lista de tablas xi

Glosario de términos xii

Capítulo 1. Introducción 1

1.1 Planteamiento del Problema 4

1.2 Antecedentes y justificación 5

1.3 Hipótesis 8

1.4 Objetivos 8

1.4.1 Objetivo general 8

1.4.2 Objetivos específicos 8

1.5 Estructura de la tesis 8

Capítulo 2. Marco teórico 10

2.1 Evolución de 4G a 6G: 10

2.1.1 *Network Slice* 12

2.1.2 Núcleo de red 5G 14

2.1.3 Función de Exposición de Red (NEF) 23

2.2 Acuerdos de nivel de servicio (SLA) 25

2.3 Tecnologías de la Web3 27

2.3.1 Blockchain 29

2.3.2 Contratos inteligentes (*Smart Contract*) 35

2.3.3 Monedero digital 37

2.3.4 Oráculos 37
2.3.5 Sistema de archivos interplanetario IPFS 39
2.4 BC y SC en la arquitectura SBA: 41
2.5 Conclusiones 44
Capítulo 3. Propuesta de arquitectura para gestión de SLA de NS 46
3.1 Propuesta de arquitectura para gestión de SLA de NS aplicando BC y SC 46
3.2 Aplicación distribuida (dApp) 50
3.3 Papel de la red IPFS 51
3.4 Funcionalidades de los Contratos Inteligentes 53
3.5 Oráculo distribuido 54
3.6 API expuesta por los proveedores 55
3.7 Arquitectura propuesta para la automatización del ciclo de vida del SLA 56
3.7.1 Registro de proveedores 57
3.7.2 Descubrimiento de proveedores 58
3.7.3 Definición de SLA 59
3.7.4 Descubrimiento de SLA 61
3.7.5 Establecimiento de SLA 61
3.7.6 Monitorización 62
3.7.7 Terminación y cumplimiento de penalidades 64
3.8 Recomendaciones de implementación 65
3.8.1 Consideraciones para implementación de la dApp 65
3.8.2 Opciones de conexión a la Blockchain 69
3.8.3 Selección de oráculo distribuido 72
3.9 Ejemplo de implementación 74
3.10 Conclusiones 75
Capítulo 4. Validación del sistema propuesto 77
4.1 Procedimiento de validación 77

4.2 Tecnologías y herramientas 78
4.2.1 Lenguaje de programación de SC 79
4.2.2 Editor de código 81
4.2.3 Herramienta para control de versiones 82
4.2.4 Framework de SC. 83
4.3 Pruebas locales 83
4.3.1 Resultados de pruebas locales 87
4.4 Pruebas en *testnet* 89
4.4.1 Prueba de tiempo y costo de ejecución de funciones de SC 93
4.4.2 Resultados de pruebas de ejecución de funciones de SC 96
4.4.3 Prueba de tiempo de respuesta a solicitud de información externa 99
4.4.4 Resultados para prueba de solicitud de información externa 100
4.5 Discusión de resultados 101
4.5.1 Discusión de resultados de pruebas locales 102
4.5.2 Discusión de resultados de ejecución de funciones en testnet 103
4.5.3 Discusión de resultados de prueba de solicitud de información externa 104
4.6 Conclusiones 105
Capítulo 5. Conclusiones y recomendaciones 107
5.1 Contribuciones al conocimiento 108
5.2 Limitaciones de la investigación 108
5.3 Trabajo futuro 109
Literatura citada 111

Lista de Figuras

Figura Página

Figura 1. Arquitectura de la red móvil 4G (Dense Networks, 2024). 11

Figura 2. Ejemplo de *Network Slice* en 5G (Xiaoyang, 2021). 12

Figura 3. Conexiones de *Network Slice* con equipos de usuario (5gAmericas, 2016). 13

Figura 4. SBA del Core 5 [Elaboración Propia] 14

Figura 5. Interacciones de una API (RedHat, 2022) 18

Figura 6. Aplicabilidad de API REST (Baniaș et al., 2021). 22

Figura 7. NEF proveyendo servicios a terceros vía *RESTful* API (Mayer, 2018) 23

Figura 8. Ciclo de vida de un SLA (Schweizer Zürich, 2019). 25

Figura 9. Estructura de datos de Blockchain (S. Zhou et al., 2023). 29

Figura 10. Los cuatro tipos de Blockchain (S. Zhou et al., 2023) 31

Figura 11. Operación de un Oráculo en Blockchain (Beniiche, 2020). 39

Figura 12. Ejemplo de DAG [Elaboración propia]. 40

Figura 13. Propuesta de gestión de SLA de NS [Elaboración Propia]. 47

Figura 14. Relación del Core 5G con esquema de gestión de SLA [Elaboración propia]. 49

Figura 15. Composición de la dApp [Elaboración propia]. 51

Figura 16. Diagrama de secuencia de registro de proveedores [Elaboración Propia]. 57

Figura 17. Diagrama de secuencia de descubrimiento de proveedores [Elaboración Propia]. 59

Figura 18. Diagrama de secuencia de definición de SLA [Elaboración Propia]. 60

Figura 19. Diagrama de secuencia de descubrimiento de SLA [Elaboración Propia]. 60

Figura 20. Diagrama de secuencia de establecimiento del SLA [Elaboración Propia]. 62

Figura 21. Diagrama de secuencia de monitorización del SLA [Elaboración Propia]. 63

Figura 22. Diagrama de secuencia de terminación y cumplimiento de penalidades [Elaboración Propia]. 64

Figura 23. Ejemplo de implementación de esquema propuesto [Elaboración Propia]. 75

Figura 24. Tecnologías para primer escenario de simulación [Elaboración propia]. 78

Figura 25. Tecnologías para segundo escenario de simulación [Elaboración propia]. 79

Figura 26. Estructura general de un SC escrito en *Solidity* [Elaboración Propia]. 81

Figura 27. Ejemplo de *test* unitario [Elaboración Propia]. 86

Figura 28. Respuesta del *test* unitario de ejemplo [Elaboración Propia]. 86

Figura 29. Ejemplo de *test* de integración escrito en *Solidity* [Elaboración Propia]. 87

Figura 30. Resultado de la prueba de cobertura de SC [Elaboración Propia] 87

Figura 31. Resultados de *test* en BC local [Elaboración Propia]. 88

Figura 32. Configuración del esquema propuesto para pruebas en *testnet* [Elaboración Propia]. 89

Figura 33. Conexión de la dApp a la BC pública por medio de un proveedor de nodos [Elaboración propia]. 91

Figura 34. Diagrama de obtención de tiempo y costo en ejecución de funciones de SC [Elaboración Propia]. 95

Figura 35. Diagrama de caja de tarifa de transacción en USD para 5 funciones de SC en la *testnet Sepolia* [Elaboración Propia]. 97

Figura 36. Diagrama de caja para tarifas de transacción en USD para 5 funciones de SC en la *testnet Mumbai* [Elaboración Propia]. 97

Figura 37. Diagrama de caja para tiempo de ejecución de 5 funciones de SC en la *testnet Sepolia* [Elaboración Propia]. 98

Figura 38. Diagrama de caja para tiempo de ejecución de 5 funciones de SC en la *testnet Mumbai* [Elaboración Propia]. 98

Figura 39. Extracción de KPI de la API del proveedor [Elaboración Propia]. 99

Figura 40. Log de eventos en interacción SC-Oráculo en *Ethereum Sepolia* [Elaboración propia]. 101

Figura 41. Log de eventos en interacción SC-Oráculo en *Polygon Mumbai* [Elaboración propia]. 101

Lista de tablas

Tabla Página

Tabla 1. *Stack* de ejemplo para aplicaciones Descentralizadas(Alabdulwahhab, 2018)..... 28

Tabla 2. Comparación entre sistemas de gestión de SLA que usan BC y SC [Elaboración Propia]. 43

Tabla 3. Comparación entre BC pública y BC privada en su uso como *backend* [Elaboración propia]. 67

Tabla 4. Comparación entre plataformas Blockchain Públicas [Elaboración propia]. 68

Tabla 5. Comparación entre plataformas BC permisionadas [Elaboración propia]. 68

Tabla 6. Comparativa entre uso de nodos propios y uso de proveedores de nodos [Elaboración Propia]. 70

Tabla 7. Comparación de la capa gratuita de servicio de proveedores de nodo [Elaboración propia]. 71

Glosario de términos

#		
3GPP	*3rd Generation Partnership Project*	Proyecto de Asociación de Tercera Generación
4G	*4th Generation Partnership*	4ta generación
5G	*5th Generation Partnership*	5ta generación
A		
ABI	*Application Binary Interface*	Interfaz de Aplicación Binaria
AF	*Application Function*	Función de Aplicación
AMF	*Access and Mobility Management Function*	Función de Gestión de Acceso y Movilidad
API	*Application Programming Interface*	Interfaz de Programación de Aplicaciones
AUSF	*Authentication Server Function*	Función del Servidor de Autenticación
B		
BSS	*Business Support System*	Sistema de Soporte al Negocio
BFT	*Byzantine Fault Tolerance*	Tolerancia a Fallas Bizantinas
B5G	*Beyond 5G*	Más allá del 5G
BC	*Blockchain*	Cadena de Bloques
C		
CAPIF	*Common API Framework*	Marco Común de API
CDN	*Content Delivery Network*	Redes de Entrega de Contenido
CN	*Core Network*	Red Central
CSV	*Comma-Separated Value*	Valor Separado por Comas
CRUD	*Create Register Update Delete*	Crear, Leer, Actualizar, Borrar
D		
DAG	*Direct Acyclic Graph*	Grafo Acíclico Dirigido
DHT	*Distributed Hash Table*	Tabla de Hash Distribuida
DPoS	*Delegated Proof of Stake*	Prueba de Participación Delegada
E		
EVM	*Ethereum Virtual Machine*	Máquina Virtual Ethereum
EB	*Block Explorer*	Explorador de Bloques
ETSI	*European Telecommunication Standard Institute*	Instituto Europeo de Normas de Telecomunicaciones
H		
HTTP	*Hypertext Transfer Protocol*	Protocolo de Transferencia de Hipertexto
HTML	*Hypertext Markup Language*	Lenguaje de Marcado de Hipertexto

I		
IoT	*Internet of Things*	Internet de las Cosas
IP	*Internet Protocol*	Protocolo de internet
IPFS	*Inter-Planetary File System*	Sistema de Archivos Interplanetario
ITU	*International Telecommunication Union*	Unión Internacional de Telecomunicaciones
J		
JSON	*JavaScript Object Notation*	Notación de Objetos de JavaScript
K		
KPI	*Key Performance Indicator*	Indicador Clave de Rendimiento
KQI	*Key Quality Indicators*	Índices de Calidad
N		
NSSF	*Network Slice Selection Function*	Función de Selección de Segmentos de Red
NEF	*Network Exposure Function*	Función de Exposición de la Red
NF	*Network Function*	Función de Red
NFV	*Network Function Virtualization*	Virtualización de Funciones de Red
NRF	*Network Repository Function*	Función del Repositorio de la Red
NS	*Network Slices*	Segmentos de Red
NWDAF	*Network Data Analytics Function*	Función de Análisis de Datos de Red
O		
OSS	*Operation Support System*	Sistema de Soporte de Operaciones
OTT	*Over The Top*	Transmisión Libre
P		
PCF	*Policy Control Function*	Función de Control de Políticas
PoA	*Proof of Authority*	Prueba de Autoridad
PoC	*Proof of Concept*	Prueba de Concepto
PoS	*Proof of Stake*	Prueba de Participación
PoW	*Proof of Work*	Prueba de Trabajo
Q		
QoS	*Quality of Service*	Calidad de Servicio
QoE	*Quality of Experience*	Calidad de Experiencia
R		
REST	*Representational State Transfer*	Transferencia de Estado Representacional
RPC	*Remote Procedure Call*	Llamada de Procedimiento Remoto
RAN	*Radio Access Network*	Red de Acceso por Radio

S

SBA	*Service-Based Architecture*	Arquitectura Basada en Servicios
SBI	*Service-Based Interface*	Interfaz Basada en Servicios
SC	*Smart Contract*	Contratos Inteligentes
SDK	*Software Development Kit*	Kit de Desarrollo de Software
SLA	*Service Level Agreement*	Acuerdo de Nivel de Servicio
SLO	*Service Level Objectives*	Objetivos de Nivel de Servicio
SDN	*Software Defined Network*	Red Definida por Software
SOAP	*Simple Object Access Protocol*	Protocolo de Acceso Simple a Objetos

T

TEE	*Trust Execution Environment*	Entorno de Ejecución Confiable

U

UPF	*User Plane Función*	Función del Plano de Usuario
URI	*Uniform Resource Identifier*	Identificador de Recursos Uniforme
URL	*Uniform Resource Locators*	Localizador Uniforme de Recursos
UDM	*Unified Data Management*	Gestión Unificada de Datos

X

XML	*Extensible Markup Lenguaje*	Lenguaje de Marcado Extensible

Capítulo 1. Introducción

A medida que la demanda de servicios móviles y la cantidad de dispositivos conectados continúan aumentando exponencialmente, la necesidad de una arquitectura de red más flexible, escalable y eficiente se vuelve crucial. La tecnología 4G, aunque ha sido un avance significativo en comparación con las generaciones anteriores, todavía presenta limitaciones en términos de capacidad, latencia y flexibilidad para adaptarse a las crecientes demandas de los usuarios y las nuevas aplicaciones emergentes de realidad aumentada, realidad virtual, ciudades inteligentes entre otras.

En las redes de nueva generación como 5G, una arquitectura basada en servicios ofrece una solución integral para abordar estas limitaciones. Esta arquitectura se basa en la virtualización de las funciones de red, lo que permite una implementación más ágil y eficiente de nuevos servicios. Al desacoplar el hardware de las funciones de red, se puede aprovechar al máximo la infraestructura existente, al tiempo que se habilita una mayor escalabilidad y flexibilidad (Rasheed, 2016).

Los diversos servicios y aplicaciones, que se desea funcionen sobre la misma infraestructura de red, necesitan cumplir con características específicas. Para satisfacer estos requisitos de servicios diversificados surge el concepto de Segmentos de Red (NS, por las siglas del término en inglés, *Network Slices*) en 5G (Wu et al., 2021a). Esta tecnología consiste en dividir la red física en redes virtuales optimizadas para aplicaciones o servicios específicos (GSM Association, 2017), para lo que se apoya en la virtualización de funciones de red. Los operadores y proveedores de infraestructura 5G encuentran beneficios en la gestión y en la capacidad de negociar sus recursos en forma de NS, lo que da paso a un mercado para diversos interesados verticales, los cuales son sectores o instituciones que para llevar a cabo su actividad necesitan cumplir determinados requisitos de comunicación. La utilización de NS, así como su mercado se espera continúen presentes en redes más allá del 5G (B5G, por las siglas del término en inglés, *Beyond 5G)* (Ren et al., 2023), por tanto, es un aspecto fundamental concebir sistemas que hagan eficiente, confiable y automatizada su gestión.

Cuando se solicita la creación de un NS, se utiliza una plantilla que el proveedor traducirá a recursos en el segmento específico según los requisitos de la aplicación o servicio, como recursos informáticos, recursos de red, recursos de radiofrecuencia, entre otros. Una vez que se crea el segmento de extremo a extremo, se firma un Acuerdo de nivel de servicio (SLA, por las siglas del término en inglés, *Service Level Agreement*) entre proveedor y cliente (Saad et al., 2021). Como su nombre lo indica, el SLA es un acuerdo oficial entre

el proveedor de servicios y los clientes o inquilinos, en base a este se define con precisión el nivel de servicio a proporcionar (Habibi et al., 2018). El SLA se utiliza para comprobar si un servicio definido se entrega según lo contratado y ayuda a gestionar la calidad del servicio (QoS, por las siglas del término en inglés, *Quality of Service*) así como su degradación. En un SLA se plasman los requisitos de QoS, por ejemplo, ancho de banda mínimo, rendimiento mínimo y latencia máxima, sin especificar la tecnología que se utilizará para ofrecer un determinado servicio (Saad et al., 2021).

Durante el establecimiento del SLA se definen los Indicadores Claves de Rendimiento (KPI, por las siglas del término en inglés, *Key Parameter Indicators*), los cuales se acuerdan entre el proveedor y el cliente. Luego se debe comprobar el cumplimiento de dichos parámetros, para lo cual de forma tradicional se utiliza un sistema de monitoreo que recopila los datos en tiempo real sobre el rendimiento del servicio tales como el *jitter*, la latencia, entre otros. Dichos datos son almacenados para su análisis posterior, el cual se refleja en reportes de monitorización que definen el cumplimiento o no del SLA. Los proveedores en este escenario presentan un gran volumen de trabajo y costo en su operación al tener que establecer acuerdos, realizar cobros, administrar sus recursos y dar mantenimiento a la infraestructura para el monitoreo de los SLA.

El dominio de las grandes corporaciones tiende a monopolizar la forma en que se miden y determinan las métricas y KPI, esto conduce a entornos gobernados donde la clientela no puede obtener una evaluación imparcial de los SLA (Kapsoulis et al., 2021). Los proveedores pueden actuar de manera maliciosa sin notificar los incumplimientos en los SLA y debido a que administran redes virtualizadas dentro de su infraestructura, los clientes tienen poco control y visibilidad sobre el monitoreo del NS (Papageorgious et al., 2020). Por su parte, los proveedores tienen la protección de su reputación e ingresos como incentivo para ocultar o disfrazar violaciones de los SLA. Estos elementos demuestran falta de transparencia en el cumplimiento de los SLA para los clientes, lo cual representa un desafío significativo en términos de confiabilidad.

En caso de discrepancias entre cliente y proveedor con respecto a las mediciones para comprobar el servicio o por ambigüedad en el contrato, la verificación del cumplimiento del SLA dependerá de un tercero mediador para resolver la disputa. Este mediador introducirá costos adicionales, burocracia y retrasos en el proceso (Afraz et al., 2023). El mediador constituye una autoridad centralizada que también puede ser influenciada en la toma de decisiones por parte del proveedor, viéndose afectado el cliente.

La necesidad de automatizar la ejecución de los SLA en los NS es crucial para garantizar una gestión

eficiente y escalable de los servicios de red en entornos complejos y dinámicos. A medida que las redes 5G y futuras generaciones se vuelven más sofisticadas, con una gran variedad de servicios y requisitos de QoS, la automatización de los SLA se convierte en un factor clave para mantener la calidad y la confiabilidad de los servicios.

En la actualidad varias investigaciones académicas y de la industria se enfocan en la inclusión de nuevas tecnologías, como la inteligencia artificial, *Blockchain* (BC, por el término en inglés, *Blockchain*) y contratos inteligentes (SC, por las siglas del término en inglés, Smart Contract), con miras a la futura estandarización de las redes 6G. Para solventar los problemas mencionados anteriormente se podría aprovechar la propiedad de inmutabilidad de las BC y la posibilidad de desplegar SC en ellas. Estos contratos son pedazos de código que pueden ejecutar los SLA de forma automatizada, sin la participación de terceros intermediarios.

BC en su definición más simple es una estructura de datos distribuida y compartida entre los miembros de la red donde se implemente. El contenido de una base de datos BC se transmite y actualiza de manera descentralizada. Esto constituye una ventaja para la transparencia de los datos debida a la ausencia de un sistema de control centralizado (Zanzi et al., 2020). Los registros que se realicen en BC son incorruptibles y esto le da la característica de inmutabilidad.

Por otro lado, los SC son códigos ejecutables que se corren y almacenan sobre la red BC para facilitar, ejecutar y hacer cumplir un acuerdo entre partes no fiables sin la participación de un tercero de confianza (H. Zhou et al., 2018). Esto es posible gracias a la inviolabilidad del código implementado en la BC, lo que asegura que las cláusulas preestablecidas se cumplan sin necesidad de intermediarios ni posibilidad de modificaciones unilaterales una vez desplegado el contrato. El proceso de validación descentralizado por parte de los nodos de la red BC garantiza la transparencia e inmutabilidad del registro de transacciones, se elimina así el riesgo de fraude o manipulación de datos. SC permite la ejecución de funciones capaces de efectuar pagos y transacciones con criptomonedas de forma que queda registrada toda su actividad en la BC. Se utilizan criptomonedas, que son activos digitales, para aprovechar la independencia de instituciones bancarias que otorgan las operaciones con criptomonedas, esto se traduce a una gestión bancaria descentralizada.

Dada las características de estas tecnologías, incluirlas en un sistema de gestión de SLA de NS aportaría transparencia, capacidades de auditabilidad y confianza al cliente. La posibilidad de automatización con SC podría reducir la carga operativa para los proveedores, ayudar con la programación de arrendamientos

de NS y eliminar la necesidad de acudir a un tercero de confianza como autoridad centralizada. Adicionalmente, la descentralización del monitoreo y verificación del cumplimiento de los SLA podría evitar la burocracia que introduce un mediador, ya que la tecnología BC y el uso de SC inmutables garantiza la persistencia de los registros de las mediciones en forma transparente y comprueba de forma automática la ocurrencia de violaciones o incidencias en el SLA contratado.

La arquitectura de las redes 6G está en proceso de ser definidas, sin embargo, se prevé se mantenga una arquitectura basada en servicios y la utilización de la tecnología NS para gestión de recursos de red, los diversos servicios existentes y los nuevos que surjan con parámetros de QoS más exigentes. La incorporación de tecnología Blockchain en la gestión de SLA para NS tiene potencial para lograr un alto nivel de transparencia, seguridad y confiabilidad a las redes 6G. Los contratos inteligentes basados en Blockchain tienen capacidades para automatizar la negociación, implementación y monitorización de SLA entre proveedores de servicios, operadores de red y usuarios finales. Esto no solo agiliza los procesos administrativos, sino que también garantizará el cumplimiento inequívoco de los términos acordados para cada NS.

El Blockchain permite una facturación más precisa y justa basada en el uso real de recursos de red. Los datos de rendimiento y utilización de cada NS se registran con estas tecnologías de manera inmutable en la cadena de bloques, proporcionando una base confiable para la resolución de disputas y la optimización continua de los servicios. Esta transparencia fomenta la confianza entre todas las partes involucradas y facilitará la creación de modelos de negocio innovadores en el futuro ecosistema 6G.

1.1 Planteamiento del Problema

Las redes de última generación 5G/B5G han evolucionado para proporcionar una amplia gama de servicios con variedad de requisitos de QoS. Para garantizar la entrega eficiente de estos servicios, se establecen SLA que cumplan con las condiciones y garantías asociadas a cada servicio que viaja sobre un NS. Sin embargo, en los esquemas tradicionales de gestión, para los clientes existe falta de transparencia en el cumplimiento de los SLA. Los proveedores tienen un alto volumen de trabajo en su gestión que puede ser reducido y automatizado, y en caso de disputa o diferencias entre los actores del contrato se necesita acudir a una entidad de confianza, un tercero que entorpece el proceso. Estos elementos afectan la eficacia en la gestión de SLA de NS lo que pudiera mejorarse con la utilización de tecnologías como BC y SC.

En este sentido, el problema a resolver en esta investigación es:

- ✓ Suplir la carencia de esquemas de gestión de SLA en redes de nueva generación que utilizan Blockchain (BC) y Smart Contract (SC) de forma que se mejore la transparencia, seguridad, la automatización y la confianza de los usuarios, se reduzca la carga operativa de los proveedores y se elimine la necesidad de un mediador.

1.2 Antecedentes y justificación

En el entorno actual, donde los servicios de telecomunicaciones son críticos, los SC ofrecen una solución atractiva para facilitar que los proveedores cumplan con los niveles de servicio acordados. Estos contratos pueden rastrear métricas como el tiempo de actividad, el ancho de banda, la latencia y otros KPI, y desencadenar automáticamente, en caso de incumplimiento, las penalizaciones al proveedor y las compensaciones correspondientes al cliente.

Además, al estar respaldados por la tecnología BC, los SC proporcionan un registro inmutable y descentralizado de los términos del SLA y su cumplimiento, lo que aumenta la transparencia y la confianza entre las partes involucradas. Si bien aún se encuentran en etapas tempranas de adopción, los SC representan una solución prometedora para mejorar la eficiencia, la transparencia y la responsabilidad en la prestación de servicios.

La idea de utilizar SC para la gestión de SLA en escenarios de telecomunicaciones inició hace relativamente poco tiempo. Por ejemplo, De Brito Gonçalves et al., (2020) se presenta una solución para verificar de manera segura y dinámica el cumplimiento de los SLA utilizando BC y SC. La evaluación de esta solución muestra que los costos y tiempos de respuesta son aceptables, sin embargo, está limitada a una sola métrica que es la disponibilidad, y no aborda la compensación monetaria automática en caso de violaciones de SLA. Los autores proponen utilizar en trabajos futuros algún mecanismo de oráculos distribuidos, que son sistemas que permiten acceder a la información externa a la BC, para aumentar la confiabilidad en la obtención de las métricas.

En Li et al. (2022) se establece una metodología para evaluar el rendimiento y escalabilidad de BC en escenarios de redes 6G y se hace una evaluación experimental del desempeño de la BC de consorcio Quorum, donde se comprueba que con la configuración adecuada se pueden satisfacer los requisitos de

rendimiento y escalabilidad para escenarios de compartición de recursos de red, de facturación y pago de servicios. Este estudio analiza el desempeño de la BC, la cual usa el mecanismo de consenso Tolerancia a Fallas Bizantinas de Estambul (IBFT siglas del inglés *Istanbul Byzantine Fault Tolerance*), pero no considera el impacto causado al utilizar otros mecanismos de consenso ni extiende sus resultados a comparaciones al utilizar otro tipo de BC.

En Azzahra & Nugraha (2023) se plantea un sistema que automatiza los procesos de acuerdo y monitoreo de SLA entre un centro de dirección de Tecnologías de la Información, que actúa como proveedor de servicios, y las diferentes secciones de la universidad, que son los usuarios de estos. Utiliza contratos inteligentes en una red BC privada lo que permite un seguimiento de los acuerdos SLA y un monitoreo continuo del cumplimiento de los objetivos de servicio.

Se señala que a pesar de que la implementación de una BC privada conlleva costos adicionales y desafíos técnicos, el sistema demuestra ser rápido y eficiente. Sin embargo, no se debe dejar de considerar los costos y complejidades en el mantenimiento y operación de un sistema que se introducen al utilizar este tipo de BC.

Las investigaciones comentadas anteriormente no tratan específicamente la gestión de SLA de *Network Slices,* pero se puede tratar de forma similar. Por ejemplo, en Costa-Pérez et al (2023) se presenta *NSBchain* como una solución de intermediación basada en BC que interconecta al proveedor de infraestructura 5G y a los inquilinos de la red. *NSBchain* utiliza contratos inteligentes para negociar y definir acuerdos de asociación entre múltiples dominios administrativos, lo que resulta en implementaciones de servicios de extremo a extremo complejas, sin embargo, no considera la monitorización.

Esta función es posible si se agrega un oráculo en el esquema de *NSBchain* que se plantea. Dicho oráculo tiene la característica de poder acceder a información fuera de la BC y entregarla a un SC y así el SC puede verificar el cumplimiento o no del acuerdo.

Para la realización de pruebas antes de hacer despliegues de contratos inteligentes en una BC se hacen pruebas de forma local, como ocurre en el artículo Saad et al. (2021). En este estudio se realizó un experimento para evaluar el desempeño de una arquitectura de confianza basada en BC y SC para la gestión de SLA en redes 5G. La arquitectura utiliza contratos inteligentes para almacenar los SLA, monitorear las violaciones y calcular las compensaciones y penalizaciones según corresponda, sin embargo, solo utiliza dos métricas de monitoreo para los experimentos cuando el conjunto de métricas

relacionas pudiera impactar en el desempeño de dicha arquitectura, empero en este trabajo no se realiza la validación de la arquitectura propuesta en una BC real.

La implementación de BC y SC en un esquema descentralizado para el comercio de recursos de red sin depender de un corredor central es un estudio realizado por (Afraz et al., 2023) donde se indica que el uso de BC y SC permite aumentar la confianza entre los operadores al eliminar un punto único de falla. Sin embargo, no se proporciona un análisis detallado de los costos asociados con las diferentes opciones de implementación de BC para este caso de uso específico. Se destacan las ventajas del enfoque descentralizado basado en BC para el arrendamiento de segmentos de red, pero también reconoce los desafíos de escalabilidad, rendimiento y costos que deben abordarse para implementaciones a gran escala.

Estas investigaciones analizadas no obstante de mostrar la viabilidad de utilizar SC para tareas de automatización de acuerdos presentan problemas como:

- ✓ Falta de análisis del impacto según el tipo de BC y con ello el mecanismo de consenso utilizado.
- ✓ Carencia de comparación en la utilización de diferentes BC.
- ✓ No utilización de oráculos distribuidos que aumenten la confiabilidad en las métricas de monitorización.
- ✓ Carencia de integración de funciones en las que puede intervenir SC como la intermediación en la negociación, monitorización, y gestión de pagos y recompensas.
- ✓ No se practica el uso de varias de métricas de monitoreo, ni su impacto en el desempeño y en la lógica para el diseño de las propuestas que involucren SC.
- ✓ Desafíos de escalabilidad y costos en la implementación.

En este sentido esta tesis va dirigida a proponer una solución integral para la gestión automatizada de SLA para NS utilizando BC y SC de una forma segura, confiable, transparente, distribuida y sin la dependencia de terceros de manera que se aborden las limitaciones identificadas en la literatura analizada.

1.3 Hipótesis

La implementación de una solución basada en Blockchain y *Smart Contracts* en el proceso de gestión de SLA de *Network Slices* permitirá aumentar la transparencia y confiabilidad para el cliente, reducir los costos operativos y mejorar la automatización de dicho proceso.

1.4 Objetivos

1.4.1 Objetivo general

Mejorar la gestión de Acuerdos de Nivel de Servicio de *Network Slices* mediante un esquema que automatice el ciclo de vida de SLA utilizando *Blockchain* y *Smart Contract*.

1.4.2 Objetivos específicos

- ✓ Proponer un sistema de gestión de Acuerdos de Nivel de Servicio en un entorno 5G/B5G.
- ✓ Utilizar tecnologías de seguridad Blockchain y *Smart Contracts*.
- ✓ Emplear programas de simulación para verificar el cumplimiento de Acuerdos de Nivel de Servicio.
- ✓ Analizar el desempeño del sistema al utilizar diferentes *Blockchains*.

1.5 Estructura de la tesis

La estructura de este trabajo de maestría indica la metodología seguida en el desarrollo de la investigación en búsqueda de alcanzar los objetivos propuestos. Los siguientes párrafos describen la organización de los capítulos para tal fin y tienen la finalidad de guiar al lector en este desarrollo:

En el **Capítulo 2** se plantea el marco teórico que será la base para el desarrollo del trabajo. Se expone la

evolución de las redes 4G a 6G, se profundiza el concepto de NS, se muestra información relacionada con el núcleo de 5G, se aborda la importancia de las API *RESTful* en la operación del núcleo de 5G y se especifican características de la Función de Exposición de red (NEF, por las siglas del término en inglés, *Network Expusure Function*). Luego se explican detalles de los SLA en el contexto de redes 5G/B5G. En la sección siguiente se abordan las tecnologías relativas a la Web3 como BC, SC, monederos digitales, oráculos y el sistema de archivos interplanetarios (IPFS, por las siglas del término en inglés*, Inter-planetary File System*) para posteriormente mostrar como BC y SC pueden estar presentes en la gestión de SLA de NS y hacer un resumen del análisis de algunos trabajos que exploraron esta opción.

En el **Capítulo 3** se presenta una propuesta de arquitectura para la gestión de SLA de NS y se explica y justifica la utilización de cada uno de sus componentes como son la dApp, la red IPFS, varios SC, el oráculo distribuido y una API que debe exponer el proveedor de NS. Luego se comentan cuáles son las interacciones entre los componentes del esquema y se presentan diagramas que ilustran estas interacciones, las cuales constituyen procedimientos que de forma general representan el ciclo de vida del SLA. Finalmente se realizan recomendaciones de tecnologías y requerimientos que necesita la arquitectura propuesta para su correcta implementación.

En el **Capítulo 4** se plantea la validación del esquema propuesto. Se muestra el procedimiento que se lleva a cabo para realizar la validación, así como la configuración utilizada para ella. Se definen las herramientas que serán utilizadas para la realización de pruebas sobre el esquema y se exponen detalles de las pruebas locales y en *testnet* que se llevaron a cabo. Al culminar el capítulo se discuten y analizan sus resultados obtenidos.

En el **Capítulo 5** quedan plasmadas las conclusiones del trabajo, destacando las contribuciones y las limitaciones percibidas. Se comenta el trabajo futuro asociado al trabajo de investigación.

Capítulo 2. Marco teórico

En la administración de redes de última generación los SLA representan un componente fundamental. Dentro del marco de la arquitectura 5G, la tecnología de Segmentación de Redes (NS) posibilita la creación de redes virtuales adaptadas a las necesidades particulares de los usuarios, así como a los requerimientos técnicos de las aplicaciones y servicios, lo que genera una mayor complejidad en la gestión de los SLA ya que se deben manejar diferentes parámetros de QoS para cada NS. A pesar de requerir una gestión de SLA más compleja, los NS posibilitan en 5G una Arquitectura Basada en Servicios (SBA, por las siglas del término en inglés, *Service Based Arquitecture*) que ofrece un enfoque versátil y modular para la construcción y gestión de redes avanzadas. El uso de contratos inteligentes puede mejorar la gestión de los SLA tanto del punto de vista del cliente como del proveedor y se debe tener en cuenta el tipo de Blockchain utilizada para evaluar el desempeño al utilizar contratos inteligentes.

El objetivo principal de este capítulo es presentar el marco teórico que será la base para el desarrollo de la propuesta de tesis. En primer lugar, se examinará la evolución de las redes móviles desde 4G hacia 5G/B5G, haciendo especial énfasis en las características claves de esta nueva generación como la virtualización de redes mediante NS y la arquitectura del núcleo 5G basada en servicios. A continuación, se analizarán los SLA y los retos que plantea su gestión automatizada en entornos de prestación de servicios en redes 5G/B5G. En la siguiente sección se introducirán algunas tecnologías de la Web3, incluyendo BC, SC, monederos de criptomonedas, oráculos e IPFS, y además se estudiarán sus fundamentos técnicos y algunas aplicaciones. Finalmente, se explorará cómo la combinación de las capacidades de las redes de nueva generación y las tecnologías Web3 puede aprovecharse para hacer disponibles nuevos modelos de negocio y esquemas de gestión y facturación automáticos para servicios de redes de 5G/B5G.

2.1 Evolución de 4G a 6G:

La red 4G está basada en una arquitectura rígida donde el plano de control, el plano de usuario, el hardware y software como entidades de red están estrechamente acopladas. Esta arquitectura es el fruto de un enfoque donde todos los servicios se tratan de forma similar. Como resultado, es difícil implementar funciones y servicios nuevos debido a que se depende del proveedor, y asignar de manera flexible y rentable los recursos necesarios para hacer frente a los nuevos requisitos de servicio y los volúmenes de

datos asociados (Sama et al., 2016). En la Figura 1 se observa la arquitectura de la red 4G.

La red 4G no solo necesita ofrecer una mayor capacidad de red y velocidad de transmisión de datos, también debe proporcionar, desde el diseño una mayor QoS y calidad de experiencia (QoE, por las siglas del término en inglés, *Quality of Experience*) (M. Yang et al., 2016). QoS se centra en los recursos de la red y refleja el grado de concordancia entre los requisitos del servicio-aplicación y la provisión de recursos de red, cuyas métricas suelen ser varios parámetros de red, como ancho de banda, latencia y *jitter* (Khanjari et al., 2013), QoE por otro lado va dirigido a reflejar la satisfacción de los usuarios finales hacia los servicios ofrecidos.

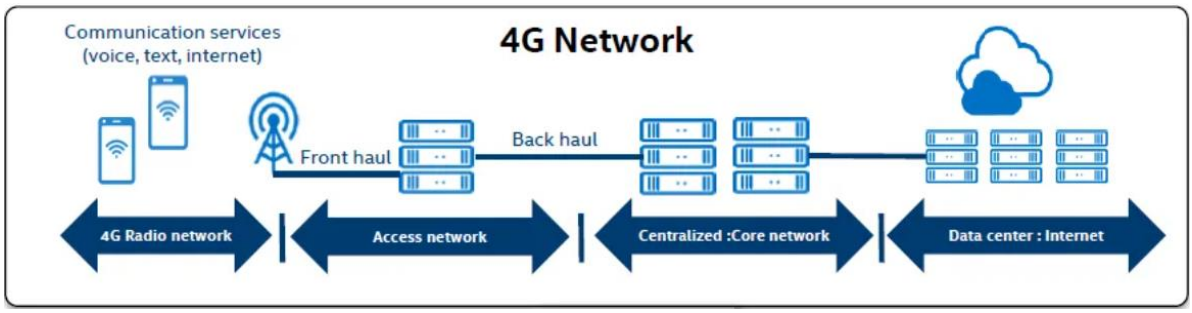

Figura 1. Arquitectura de la red móvil 4G (Dense Networks, 2024).

Las tecnologías de Redes Definidas por Software (SDN, por las siglas del término en inglés, *Software Defined Network)* y Virtualización de Funciones de Red (NFV, por las siglas del término en inglés, *Network Function Virtualization*) en el contexto de la transición hacia la tecnología 5G posibilitan enfrentar las limitaciones inherentes de las redes 4G. Permiten una arquitectura de red orientada a servicios, superando así las dificultades que enfrentan las redes 4G para adaptarse a la rápida proliferación de servicios móviles. Al introducir un plano de control lógicamente centralizado, estas tecnologías simplifican y separan el plano de datos, facilitando la gestión de los diversos requisitos del servicio móvil(Gonzalez-Franco et al., 2023).

Es crucial destacar que SDN y NFV mejoran significativamente tanto la QoS como la QoE de extremo a extremo, aspectos que las redes 4G no pueden garantizar debido a su diseño orientado a la voz. La flexibilidad y programabilidad ofrecidas por SDN y NFV, mediante interfaces abiertas y configurables, permiten a los proveedores de servicios adaptarse de manera dinámica a los requisitos cambiantes del servicio (M. Yang et al., 2016). Estas características no solo abordan las limitaciones de las redes 4G, sino que también sientan las bases para una infraestructura móvil 5G más eficiente.

2.1.1 *Network Slice*

El concepto Segmentos de Red (NS, por las siglas del término en inglés, *Network Slices*) surge como una solución para satisfacer los requisitos de servicios diversificados, dividiendo la red física en redes virtuales optimizadas para aplicaciones o servicios específicos (Wu et al., 2021). El NS permite a los operadores particionar las redes de forma estructurada, elástica, escalable y automatizada para reducir el costo total, disminuir el consumo de energía y simplificar las funciones de red (GSM Association, 2017). Los operadores encuentran beneficios en la eficiencia de recursos y en la capacidad de negociar estos recursos en forma de NS, creando un mercado en crecimiento para diversos interesados verticales, que se espera continúe en redes posteriores a 5G.

NS permite que los elementos y funciones de la red se configuren y reutilicen fácilmente en cada segmento de red para cumplir con requisitos específicos. La implementación de la división de red se concibe como una característica de extremo a extremo que incluye la red central y la Red de Acceso por Radio (RAN, por las siglas del término en inglés, *Radio Access Network*). Cada segmento puede tener su propia arquitectura de red, mecanismo de ingeniería y aprovisionamiento (Ramos et al., 2021). En la Figura 2 se puede observar cómo se asignan determinados recursos para cada tipo de servicio teniendo en cuenta las características específicas del servicio.

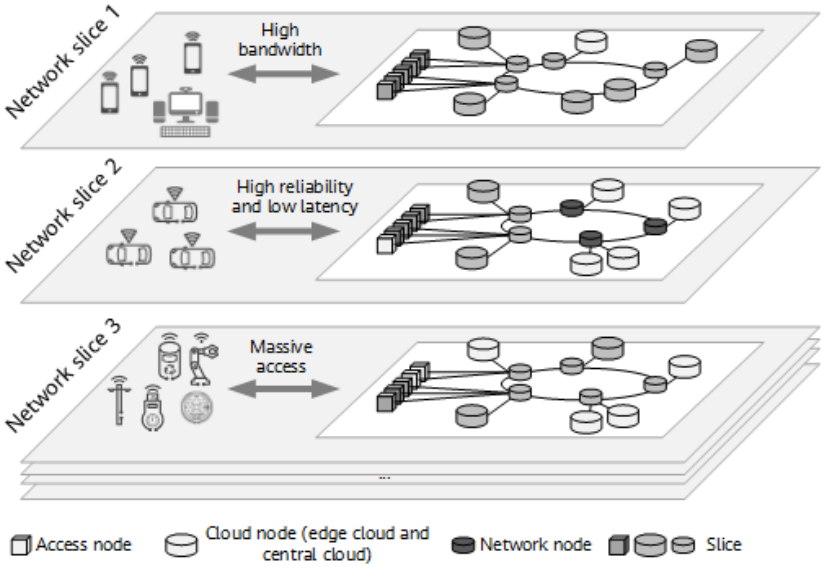

Figura 2. Ejemplo de *Network Slice* en 5G (Xiaoyang, 2021).

La combinación de tecnologías de nube con las capacidades de redes definidas por software (SDN) y virtualización de funciones de red (NFV) proporciona las herramientas necesarias para permitir la división

de redes ya que permiten el uso de recursos físicos y virtuales para crear NS para cada tipo de servicio. Se prevé que esta tendencia continuará, donde las tecnologías de virtualización se aplicarán en RAN y servicios y dispositivos portátiles (5gAmericas, 2016).

Un NS es una red autónoma dentro de una red más grande, que cuenta con sus propios recursos virtuales, topología, patrones de tráfico y reglas de aprovisionamiento. Esto permite a los usuarios gestionar su propia infraestructura de red dedicada de acuerdo con sus necesidades puntuales. Estas redes lógicas se establecen y gestionan de forma flexible para adaptarse a diferentes servicios y requisitos de comunicación de los usuarios. La segmentación de redes es considerada un componente clave en los sistemas 5G/B5G, ya que permite que las redes sean flexibles, ágiles y escalables para responder rápidamente a los cambiantes requerimientos empresariales (Wu et al., 2021b) .

La arquitectura de división de red 5G/B5G se compone de segmentos de acceso, ya sean inalámbricos o fijos. Estos se conectan a segmentos de Red Central (CN, por las siglas del término en inglés, *Core Network*) a través de una función de selección que los integra en un segmento de red completo que abarca el acceso y la CN. Esta función de selección enruta las comunicaciones hacia el segmento de CN apropiado, diseñado para ofrecer servicios específicos que satisfacen los diferentes requerimientos de las aplicaciones y tipos de comunicación (5gAmericas, 2016).

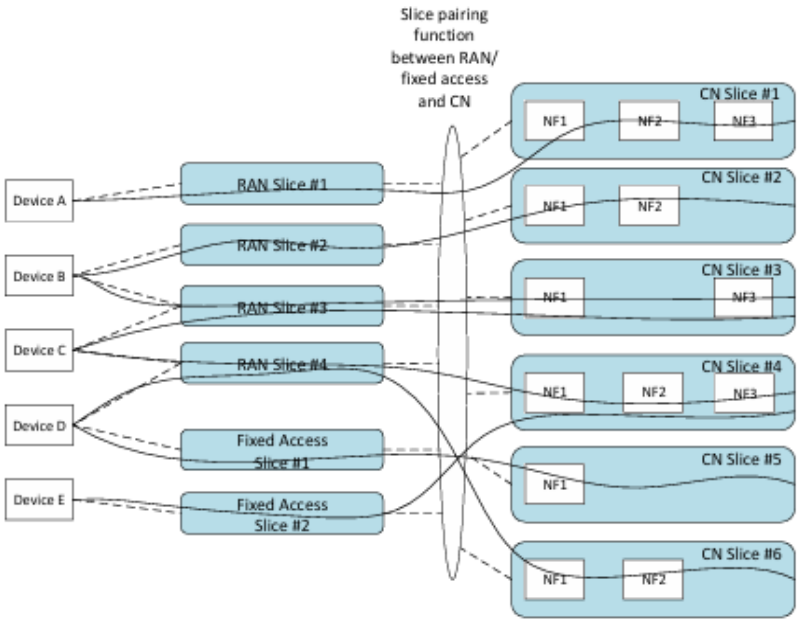

Figura 3. Conexiones de *Network Slice* con equipos de usuario (5gAmericas, 2016).

Cada segmento de CN se construye a partir de un conjunto de funciones de red (NF, por las siglas del

término en inglés, *Network Funcitons*). Es importante destacar que algunas NF pueden ser compartidas entre múltiples segmentos, mientras que otras están dedicadas a un segmento específico. El mapeo entre dispositivos, segmentos de acceso y segmentos de CN puede tener relaciones de 1 a 1, o bien de 1 a muchos y muchos a muchos, como se ilustra en la Figura 3. Por ejemplo, un dispositivo podría utilizar varios segmentos de acceso, y un segmento de acceso podría estar conectado a múltiples segmentos de CN (5gAmericas, 2016).

2.1.2 Núcleo de red 5G

Tradicionalmente, los CN se han diseñado como una arquitectura única que cumple múltiples propósitos, aborda una variedad de requisitos y admite compatibilidad e interoperabilidad con versiones anteriores. Este enfoque integrado ha mantenido los costos en un nivel razonable al proporcionar todas las funciones a través de un conjunto de nodos integrados verticalmente. Sin embargo, la evolución tecnológica hacia la virtualización, NFV, SDN y la automatización y orquestación avanzada está permitiendo la construcción de redes de una manera más escalable, flexible y dinámica. Estas capacidades permiten a los diseñadores de redes actuales adoptar un enfoque radicalmente diferente para el núcleo, brindando mayores posibilidades para soluciones optimizadas y personalizadas (5gAmericas, 2016), tal es el caso del núcleo de red de 5G.

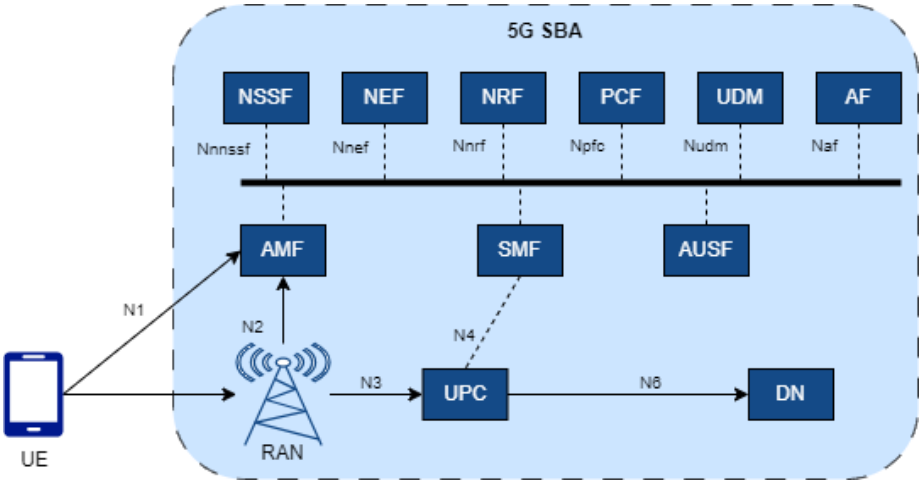

Figura 4. SBA del Core 5 [Elaboración Propia]

La SBA se puede observar en la Figura 4, esta figura se confeccionó en base a la información que aparecen en (3GPP, 2019). SBA está definida para las redes 5G por el Proyecto de Asociación de 3ra Generación (3GPP, por las siglas del término en inglés, *Third Generation Partnership Project*). Esta arquitectura

modular está compuesta por un conjunto de funciones de red interconectadas, encargadas de gestionar los planos de usuario y control de la red. La premisa fundamental de la SBA es descomponer las funcionalidades tradicionales de las entidades de red en servicios independientes y expuestos a través de Interfaces Basadas en Servicios (SBI, por las siglas del término en inglés, *Service Based Interfaces*). Estos servicios se ofrecen a otras funciones de red utilizando un bus de mensajería SBI que implementa API *RESTful* sobre el protocolo HTTP/2 (3GPP, 2019).

El CN de 5G es una parte fundamental de la arquitectura de la red 5G, diseñada para soportar las demandas de conectividad avanzada y de alto rendimiento. A continuación, se muestran las responsabilidades de algunas funciones virtuales que conforman el CN de 5G (3GPP, 2019):

- Función de Gestión de Acceso y Movilidad (AMF, por las siglas del término en inglés, *Access and Mobility Management Function*):
  - Gestiona las solicitudes de conexión y autenticación de los dispositivos.
  - Controla la movilidad de los usuarios entre diferentes estaciones base.
  - Proporciona información de autenticación y autorización.
- Función de Gestión de Sesiones (SMF, por las siglas del término en inglés, *Session Management Function*):
  - Administra las sesiones de datos de los usuarios.
  - Asigna direcciones IP a los dispositivos.
  - Controla el establecimiento, modificación y liberación de los túneles de datos.
- Función del Plano de Usuario (UPF, por las siglas del término en inglés, *User Plane Function*):
  - Maneja el tráfico de datos del usuario y su encaminamiento a través de la red.
  - Realiza funciones de QoS y políticas de tráfico.

- Interactúa con las redes externas y los servicios de internet.

- Función de Control de Políticas (PCF, por las siglas del término en inglés, *Policy Control Function*):
  - Define y aplica políticas de control de QoS.
  - Gestiona las políticas de control de suscripción y servicios.
  - Coordina las políticas de red entre las diferentes funciones.

- Gestión Unificada de Datos (UDM, por las siglas del término en inglés, *Unified Data Management*):
  - Almacena y gestiona la información de suscripción y perfil de usuario.
  - Proporciona datos de usuario a otras funciones de la red bajo demanda.

- Función del Servidor de Autenticación (AUSF, por las siglas del término en inglés, *Authentication Server Function*):
  - Realiza la autenticación de los usuarios y dispositivos.
  - Trabaja en conjunto con AMF para autenticar a los suscriptores.

- Función de Selección de Segmentos de Red (NSSF, por las siglas del término en inglés, *Network Slice Selection Function*):
  - Asigna y gestiona los segmentos de red, que son particiones lógicas de la infraestructura de red para distintos tipos de servicios.
  - Facilita la diferenciación de servicios y aplicaciones con diferentes requisitos de rendimiento.

- Función de Exposición de la Red (NEF, por las siglas del término en inglés, *Network Exposure Function*):
  - Permite la exposición de las capacidades de la red a aplicaciones externas a través de API.

- Facilita la integración con aplicaciones de terceros y servicios de transmisión libre (OTT, por las siglas del término en inglés*, Over-The-Top*).

- Función del Repositorio de la Red Función de (NRF, por las siglas del término en inglés, *Network Repository Function*):

  - Mantiene un registro de todas las funciones de red disponibles y sus capacidades.

  - Facilita el descubrimiento y la selección de funciones de red por otras funciones de red.

- Función de Aplicación (AF, por las siglas del término en inglés, *Application Function*):

  - Permite la interacción entre las aplicaciones y la red 5G.

  - Proporciona información de control y servicios específicos a las aplicaciones.

En 5G las funciones virtuales de red están interconectadas y colaboran para brindar una plataforma sólida, adaptable y escalable. Confirman una arquitectura diseñada para respaldar una amplia gama de aplicaciones y casos de uso en el ecosistema 5G, incluido el Internet de las Cosas, comunicaciones de alta velocidad y servicios críticos que requieren baja latencia. Una característica clave de esta arquitectura basada en servicios es la interconexión de todas sus funciones de red mediante API *RESTful*, lo que permite que las diferentes funciones puedan intercambiar datos a través de solicitudes HTTP. Su enfoque abierto basado en API facilita la integración de API de terceros en el bus SBI, siempre y cuando se cumplan los protocolos de seguridad establecidos. Esta aproximación modular e interoperable brinda una gran flexibilidad y facilita la escalabilidad de la red 5G, lo que concede adaptarse a las demandas cambiantes y requisitos de los usuarios y las aplicaciones emergentes (Gonzalez-Franco et al., 2023).

2.1.2.1 Interfaz de Programación de Aplicaciones (API)

Una interfaz de programación de aplicaciones (API, por las siglas del término en inglés, *Application Programming Interface*) permite la comunicación entre aplicaciones para integrar un software secundario en el código fuente de la aplicación principal. Estas piezas de aplicaciones se comunican entre sí en el idioma que ambas entienden y, si es necesario, a través de una red (Jacobson et al., 2011). En otras

palabras, es un conjunto de definiciones, protocolos y herramientas para desarrollar e integrar aplicaciones de software. Actúa como una capa intermedia que permite la comunicación e interacción entre diferentes componentes de software.

Una API, permite a entidades comerciales, militares o privadas poner los datos y la funcionalidad de sus aplicaciones o sistemas a disposición de desarrolladores externos, socios comerciales y departamentos internos dentro de sus propias organizaciones. El uso de una interfaz definida permite que servicios y productos interactúen entre sí y se beneficien de la información y características de cada uno. Esta interfaz es manipulada y programada por los desarrolladores para interactuar con otro software, servicios o sistemas. Los desarrolladores no están obligados a comprender cómo está desarrollada una API específica, y tampoco lo son los usuarios finales del software (Nadim Iqbal, 2023).

Una API actúa como una capa de abstracción que define cómo se deben estructurar las solicitudes y respuestas, qué operaciones se pueden realizar y cómo se deben formatear los datos enviados y recibidos. Al seguir las especificaciones de la API, sistemas diferentes pueden interoperar sin necesidad de conocer los detalles internos de cada uno (RedHat, 2022). Generalmente su arquitectura se describe en términos de cliente y servidor. El cliente es la aplicación que envía la solicitud, y el servidor es la aplicación que envía la respuesta. En la Figura 5 se muestra la interacción de una API donde los servidores son los llamados *Backend systems* y los clientes son otras aplicaciones, por ejemplo, aplicaciones de sensores de Internet de las cosas (IoT, por las siglas del término en inglés, *Internet of Things*) o aplicaciones de dispositivos móviles.

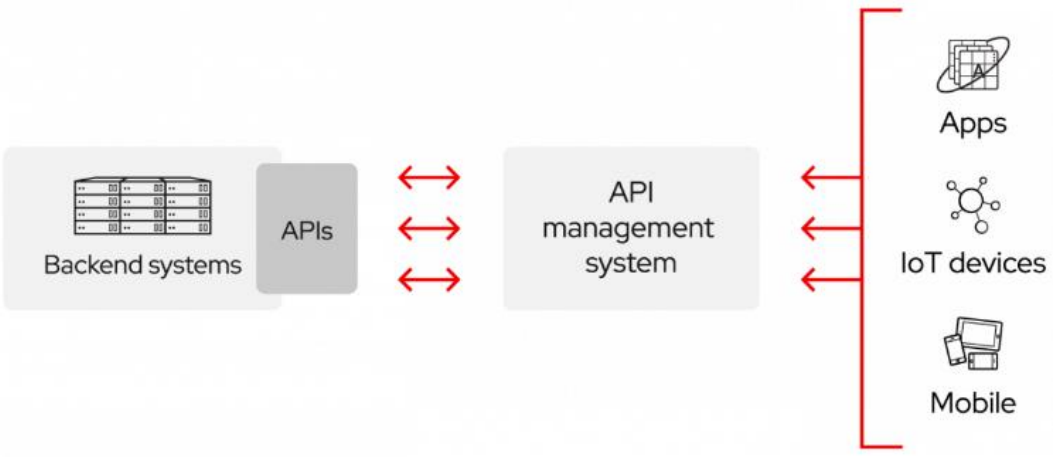

Figura 5. Interacciones de una API (RedHat, 2022)

Las API se utilizan ampliamente en el desarrollo de software moderno, se utilizan prácticamente en

cualquier entorno ya sea Web, móvil, etc. A continuación, se exponen de forma resumida algunas funciones de una API (Jacobson et al., 2011):

- Integración de servicios: permiten que diferentes aplicaciones o servicios se comuniquen y compartan datos y funcionalidades entre sí, facilitando la integración de sistemas heterogéneos.

- Acceso a datos y funcionalidades: pueden proporcionar acceso controlado a datos y funcionalidades específicas de un sistema, sin necesidad de exponer todo el código fuente o la lógica interna.

- Desarrollo de aplicaciones: facilitan el desarrollo de aplicaciones al permitir que los desarrolladores reutilicen funcionalidades existentes en lugar de tener que construirlas desde cero.

- Escalabilidad y modularidad: promueven el modularidad y la separación de preocupaciones al dividir las funcionalidades en componentes independientes, lo que facilita la escalabilidad y el mantenimiento del software.

2.1.2.2 Protocolo de Transferencia de Hipertexto (HTTP)

Entre las opciones de protocolos que tienen las API para su funcionamiento está el Protocolo de Transferencia de Hipertexto (HTTP, por las siglas del término en inglés, *Hipertext Transfer Protocol*) como protocolo de comunicación para permitir la interacción entre sistemas informáticos a través de la Web. HTTP proporciona un conjunto de reglas para la solicitud y respuesta de datos entre clientes y servidores, lo que lo hace ideal para la implementación de API basadas en la Web. Es uno de los protocolos más omnipresentes en Internet. También es uno de los pocos protocolos que cierra la brecha entre los grupos de desarrollo de aplicaciones y de redes, y contiene información que se utiliza tanto en la entrega como en el desarrollo de aplicaciones Web (Chang, 2015) .

HTTP es el protocolo que se utiliza para la transferencia de información a través de Internet. Permite la comunicación entre clientes y servidores. A continuación, se muestran los aspectos fundamentales de HTTP (Mogul, 2002):

- HTTP sigue un modelo cliente-servidor, donde el cliente realiza una solicitud y el servidor responde con los recursos solicitados por ejemplo páginas Web, imágenes, archivos o acciones.

- La comunicación en HTTP se basa en un ciclo de solicitud-respuesta. El cliente envía una solicitud HTTP al servidor, y el servidor responde con una respuesta HTTP.

- Define varios métodos de solicitud, siendo los más comunes GET (para obtener recursos), POST (para enviar datos), PUT (para actualizar recursos) y DELETE (para eliminar recursos).

- Tanto las solicitudes como las respuestas HTTP incluyen cabeceras, que son metadatos que proporcionan información adicional sobre la solicitud o respuesta, como el tipo de contenido, la codificación, la caché, etc.

- Las respuestas HTTP incluyen códigos de estado que indican el resultado de la solicitud. Algunos códigos comunes son: 200 OK, que representa una solicitud exitosa, 404 *Not Found*, que significa que el recurso no fue encontrado, y 500 *Internal Server Error*. que quiere decir que hubo un error en el servidor.

- Un Localizador Uniforme de Recursos (URL, por las siglas del término en inglés, *Uniform Resource Locators*) se utilizan para identificar los recursos en Internet. En HTTP, las URL especifican la ubicación del recurso solicitado en el servidor.

- HTTP utiliza *cookies* para mantener el estado entre solicitudes. Las *cookies* son pequeños fragmentos de datos que el servidor envía al cliente y que el cliente devuelve en solicitudes posteriores.

- Para una comunicación segura, se utiliza HTTPS, que es HTTP sobre una capa de seguridad de la capa de transporte pues por sí solo no utiliza cifrado de datos.

- HTTP es un protocolo sin estado, lo que significa que cada solicitud es independiente y no tiene conocimiento de las solicitudes anteriores. Esto se aborda mediante el uso de *cookies*, sesiones y otros mecanismos.

- HTTP permite el almacenamiento en caché de recursos, lo que mejora el rendimiento al reducir la

necesidad de descargar los mismos recursos repetidamente.

2.1.2.3 API *RESTful*

La transferencia de estado representacional (REST, por las siglas del término en inglés, *Representational State Transfer*) es una arquitectura de software que establece pautas sobre cómo debe operar una API, aunque no se trata de un estándar formal. Originalmente, REST se concibió como un marco de referencia para gestionar la comunicación en redes complejas como Internet. Ofrece facilidad de implementación y modificación, brindando visibilidad y portabilidad entre plataformas para cualquier sistema de API. Los desarrolladores pueden diseñar API siguiendo diferentes arquitecturas, y aquellas que cumplen con los principios arquitectónicos de REST se denominan API REST o API *RESTful* (Amazon Web Services, 2024) .

REST establece reglas que garantizan la comunicación entre cliente y servidor de forma eficiente escalable y mantenible. REST plantea que debe utilizarse un formato común para realizar esta comunicación el cual puede ser JSON, XML o incluso HTML, siempre siguiendo el protocolo HTTP (Baniaș et al., 2021). El tipo de formato utilizado se determina en el encabezado HTTP.

En REST se define la forma en que se realiza una solicitud en la arquitectura cliente servidor. Esta solicitud requiere dos elementos esenciales: un método HTTP y un Identificador de Recursos Uniforme (URI, por las siglas del término en inglés, *Uniform Resource Identifier*). En la Figura 6 se muestra la operación de una API REST. Se define que los datos están representados como recursos, a los cuales se tiene acceso mediante un URI único (Massé, 2011). Estos recursos pueden ser objetos, colecciones de objetos o incluso acciones. Las operaciones de CRUD (Crear, Leer, Actualizar, Eliminar) se realizan mediante los métodos HTTP estándar:

- GET: para obtener un recurso o lista de recursos.
- POST: para crear un nuevo recurso.
- PUT: para actualizar un recurso existente.
- DELETE: para eliminar un recurso.

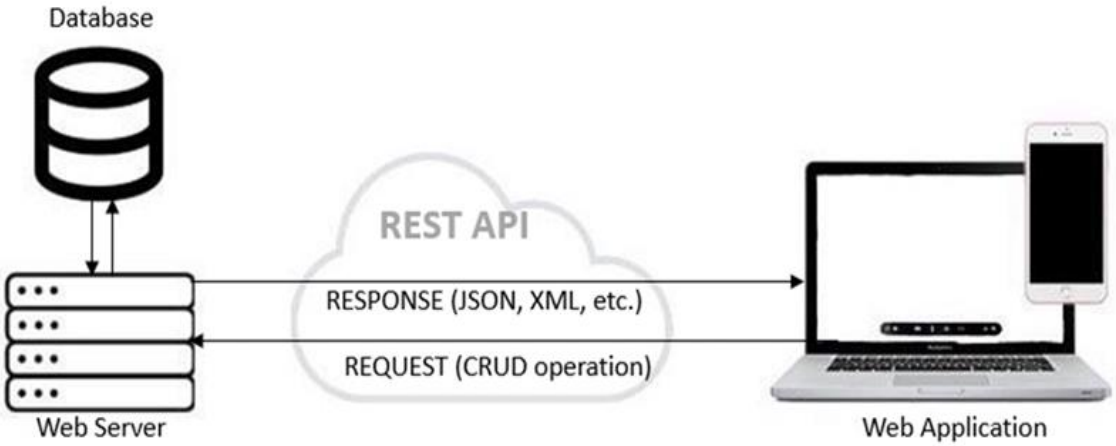

Figura 6. Aplicabilidad de API REST (Baniaș et al., 2021).

La utilización de esta arquitectura trae muchas ventajas dadas sus características. Cada solicitud HTTP debe contener toda la información necesaria para ser procesada por el servidor, sin depender del estado previo del servidor, por eso REST se considera *stateless*, pues cada solicitud por si sola es autodescriptiva y no depende de una solicitud anterior. Esto permite que los servicios *RESTful* sean escalables y puedan manejar múltiples solicitudes simultáneas. Las respuestas del servidor pueden ser cacheadas por el cliente o por intermediarios que pueden ser *proxies*, Redes de Entrega de Contenido (CDN, por las siglas del término en inglés, *Content Delivery Network*), etc., para mejorar el rendimiento y reducir la carga en el servidor. Existe una interfaz uniforme para interactuar con los recursos, lo que facilita la comprensión y la interoperabilidad entre diferentes sistemas (Pautasso et al., 2008) .

Un estilo de arquitectura anterior a REST es el Protocolo de Acceso Simple a Objetos (SOAP, por las siglas del término en inglés, *Simple Object Access Protocol*), el cual se centra en un modelo de comunicación basado en mensajes (XML, por las siglas del término en inglés, *Extensible Markup Language*), lo que permite la transferencia de datos estructurados entre sistemas heterogéneos (Nadim Iqbal, 2023). A medida que la Web evolucionaba y las necesidades de desarrollo de aplicaciones cambiaban, algunas limitaciones de SOAP se hicieron evidentes. La estructura pesada de los mensajes XML y la necesidad de implementar estándares de seguridad y protocolos adicionales hace que SOAP sea más difícil de adoptar y entender para desarrolladores nuevos o menos experimentados. Además, SOAP tiende a tener una sobrecarga de rendimiento debido al procesamiento adicional requerido para interpretar los datos XML. Esto resulta en un rendimiento inferior en comparación con REST, especialmente en entornos con ancho de banda limitado o alta concurrencia. Mientras que REST utiliza métodos HTTP estándar para realizar operaciones CRUD, SOAP requiere la definición explícita de servicios y operaciones (Tihomirovs & Grabis, 2017).

REST, una arquitectura más simple y flexible, se ha convertido en la preferida para muchas aplicaciones modernas debido a varias desventajas clave de SOAP, tanto así que las funciones virtuales de red del núcleo de 5G utilizan API *RESTful* para su comunicación. El concepto de API *RESTful* se inserta en el núcleo de 5G lo cual facilita la interacción entre los servicios y las aplicaciones en la red, y permite así una comunicación eficiente y escalable mediante la transferencia de datos a través de interfaces uniformes y basadas en estándares Web. Estas API proporcionan una arquitectura flexible y adaptable que permite a los desarrolladores crear aplicaciones innovadoras y aprovechar al máximo las capacidades de la red 5G.

2.1.3 Función de Exposición de Red (NEF)

La NEF desempeña un papel crucial en la monitorización de parámetros de los NS en las redes 5G. Constituye un punto de acceso seguro para que las aplicaciones externas y los proveedores de servicios puedan interactuar con las funciones de red del núcleo 5G. Gracias a su capacidad de exponer API *RESTful*, facilita el monitoreo y la recopilación de información sobre el rendimiento, la calidad de servicio y otros parámetros clave de los NS. Mediante las API que expone, las aplicaciones y los sistemas de gestión pueden solicitar y recibir datos en tiempo real sobre el estado de los NS (Mayer, 2018). Esto incluye métricas como el ancho de banda disponible, la latencia, la tasa de pérdida de paquetes, el número de usuarios conectados, entre otros indicadores relevantes. En la Figura 7 se observa como NEF provee servicios a terceros vía *RESTful* API.

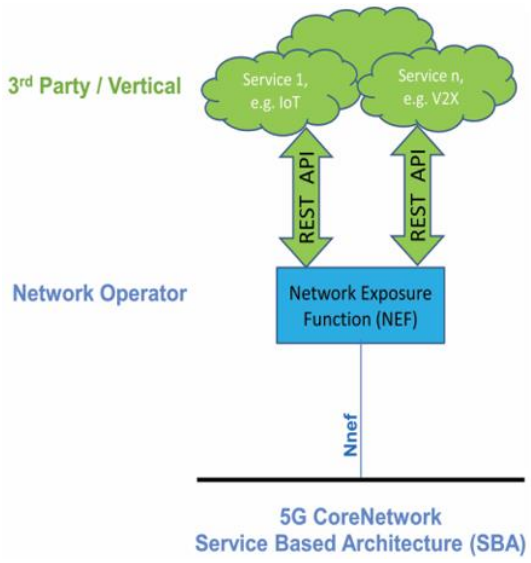

Figura 7. NEF proveyendo servicios a terceros vía *RESTful* API (Mayer, 2018).

Un proceso industrial vertical es una estrategia empresarial en la que una compañía controla múltiples etapas de la cadena de valor dentro de una misma industria, abarcando desde la obtención de materias primas o recursos iniciales hasta la entrega del producto final o servicio al consumidor. Cuando los operadores de telecomunicaciones buscan integrar la tecnología 5G con procesos industriales verticales, es crucial compartir datos sobre las capacidades de la red tanto con aplicaciones internas como externas. La estandarización de las interfaces que permiten este intercambio de información facilita la integración para los clientes industriales, especialmente aquellos que tienen operaciones a nivel internacional y que deben relacionarse con múltiples operadores de telecomunicaciones (Telecommunication Engineering Centre Khurshid Lal Bhawan, 2021).

Los operadores de telecomunicaciones y proveedores de servicios utilizan la interfaz SBI para obtener información sobre el rendimiento de los NS que arriendan a sus clientes. Mediante una API, los operadores pueden exponer a sus clientes los KPI que describen el comportamiento de los NS contratados. Los clientes tienen puntos de acceso, a través de la API, a los que se conectan para extraer los datos necesarios y comprobar el cumplimiento de los SLA para cada NS previamente establecido entre cliente y proveedor.

Anteriormente, cada operador diseñaba sus propias API de forma independiente, lo que dificultaba la interoperabilidad y la integración con diferentes proveedores. Sin embargo, el 3GPP (Proyecto Asociación de Tercera Generación) en su versión 16 definió un marco de trabajo común de api (CAPIF, por las siglas del término en inglés*, Common API Framework*) con el objetivo de estandarizar y facilitar la exposición de las capacidades del núcleo de red 5G hacia aplicaciones y servicios de diversas industrias verticales (Charismiadis et al., 2023).

CAPIF actúa como un punto de acceso único y centralizado, que permite a los operadores exponer de manera controlada y segura las funcionalidades de sus redes a clientes externos, como empresas de diferentes sectores industriales. Esto promueve la innovación y el desarrollo de nuevos casos de uso, al tiempo que simplifica la integración y la interoperabilidad entre los proveedores de servicios y sus clientes (Charismiadis et al., 2023).

Los arrendatarios de NS, al tener acceso a la información relativa al segmento de red contratado, pueden concebir un sistema que se plantee verificar el cumplimiento o no de un SLA de NS, el cual debe basarse en las métricas de monitoreo del NS correspondiente, para, según los parámetros de QoS definidos para el servicio contratado, determinar si se están cumpliendo con los umbrales para cada métrica y detectar

la posibilidad de violaciones en la prestación del servicio. Sin embargo, la existencia de mecanismos diferentes para la comprobación del SLA por parte de cliente y proveedor puede dar lugar a discrepancias.

2.2 Acuerdos de nivel de servicio (SLA)

Un Acuerdo de Nivel de Servicio (SLA) es un contrato que estable entre un proveedor de servicios y un cliente. Es un documento legal y detallado que incluye las funciones y responsabilidades para cumplir por las partes implicadas, como el periodo de duración, la calidad del servicio, y las penalizaciones (Jin et al., 2002).

Los Acuerdos de Nivel de Servicio (SLA) permiten identificar y definir claramente las necesidades y requerimientos del cliente en cuanto al servicio que recibirá. Por otro lado, también establecen y controlan las expectativas del cliente en relación con las capacidades y limitaciones del proveedor del servicio. Los SLA deben abordar asuntos complejos de manera simplificada, proporcionando un marco de entendimiento común entre el proveedor y el cliente. De esta forma, se reducen las áreas potenciales de conflicto o malentendidos.

Figura 8. Ciclo de vida de un SLA (Schweizer Zürich, 2019).

Los SLA favorecen el diálogo y la comunicación abierta entre las partes involucradas en caso de presentarse alguna disputa o incumplimiento, facilitando la resolución de problemas. También constituyen un punto de referencia para el proceso de mejora continua, ya que, poder medir adecuadamente los niveles de servicio es el primer paso para mejorarlos y cumplir con los índices de desempeño (KPI), e índices de calidad (KQI, por las siglas del término en inglés, *Key Quality Indicators*), etc. (Kapassa et al., 2018).

En el ámbito de redes de telecomunicaciones, la gestión de nivel de servicios (SLM, por sus siglas en inglés, *Service Level Management*) considera los objetivos de nivel de servicio (SLO, por las siglas del término en inglés, *Service Level Objetives*) que se basan en parámetros de QoS. En las redes 5G basadas en NS, cada métrica está relacionada con la calidad del servicio, como por ejemplo latencia, retardo, velocidad de datos, capacidad, rendimiento, movilidad, seguridad, consumo de energía, densidad de conexión, tiempo de respuesta, nivel de servicio, etc., definidas y estandarizadas por organizaciones, como la Unión Internacional de Telecomunicaciones (UIT, por las siglas del término en inglés, *International Telecommunication Union*), el ETSI (por sus siglas en inglés *European Telecommunications Standards Institute*), entre otras (Habibi et al., 2018).

Una vez establecido los SLO, se plasman en el SLA cuyas métricas deben ser monitoreadas bajo los diferentes umbrales que serán específicos según el servicio para el cual este destinado el NS. En (Schweizer Zürich, 2019) se plantea el ciclo de vida de un SLA dividido en varias fases, que se muestra en la Figura 8. Este servirá como de guía para automatizar la ejecución de SLA en la propuesta.

A continuación, se define cada fase de este ciclo de vida:

- Descubrimiento de Proveedor de Servicios: La identificación del Proveedor de Servicios depende de los requisitos del cliente, por ejemplo, presupuesto definido, ubicación geográfica, mecanismos de seguridad, preocupaciones de privacidad.

- Definir SLA: Después de encontrar un proveedor, se inicia un proceso de negociación donde ambas partes buscan llegar a un acuerdo sobre los SLO, así como definir las penalizaciones asociadas y costos involucrados. Es esencial que ambas partes compartan un entendimiento mutuo de las expectativas del otro para garantizar que el SLA sea claro y sin ambigüedades y evitar posibles problemas cuando el acuerdo entre en vigor. En caso de que las partes no logren llegar a un acuerdo, el proceso del ciclo de vida se reinicia hasta que se alcance un consenso satisfactorio.

- Establecer Acuerdo: Esta fase consiste en definir y desarrollar la plantilla en la que tendrá lugar el SLA. Ambas partes están obligadas a los términos definidos en la fase anterior tan pronto como firmen el contrato. Los servicios definidos se implementan y están listos para su acceso y utilización por el cliente.

- Monitorear Violación del SLA: Los servicios implementados deben ser monitoreados para asegurar que el Proveedor de Servicios cumpla con los términos acordados. Ambas partes monitorean los servicios por su cuenta o confían en una solución de terceros. En cualquier caso, la solución de monitoreo debe proporcionar mediciones confiables.

- Terminar el SLA: Si no se produjo ninguna violación y el servicio se entregó según lo acordado, el SLA termina cuando expira. Dependiendo de los términos definidos en la Fase 2, también puede terminar antes. Por ejemplo, si el número de violaciones detectadas estaba por encima de un umbral definido.

- Hacer cumplir las penalizaciones por Violación del SLA: En el caso de que el Proveedor de Servicios no cumpla con los niveles de desempeño especificados, el cliente deberá ser compensado. El valor de la compensación se calcula según las penalizaciones definidas en el SLA. Estas compensaciones pueden ser en forma de moneda fija o créditos de servicio. Por ejemplo, la compañía Amazon después de que el equipo responsable haya confirmado la violación, compensa a los clientes en forma de créditos de servicio dentro de un mes de facturación (Schweizer Zürich, 2019).

Las fases del ciclo de vida de los SLA son una guía para seguir y son elementos importantes por considerar en un esquema que se plantee la automatización de dicho ciclo de vida. En el ámbito de las redes de telecomunicaciones 5G, los acuerdos de nivel de servicio para NS son fundamentales para garantizar la calidad y el rendimiento acordados con los clientes. A pesar de ello, la gestión y verificación de estos SLA suele ser un proceso complejo y propenso a disputas entre las partes involucradas. Es aquí donde las innovadoras tecnologías de la Web3 desempeñan un papel crucial, ofrece la posibilidad de construir aplicaciones descentralizadas, transparentes y confiables que automaticen el ciclo de vida de SLA de NS.

2.3 Tecnologías de la Web3

La Web 2.0 o segunda generación de internet ha sido un gran avance tecnológico permitiendo la interacción en redes sociales y una conexión a internet más dinámica e interactiva. Sin embargo, ha

generado varios problemas al estar basada en un modelo centralizado controlado por grandes empresas y proveedores de servicios, que tienen numerosos centros de datos y servidores. Entre ellos se encuentran la falta de privacidad y control sobre los datos personales, la posibilidad de censura y manipulación de contenidos, así como problemas de seguridad informática por la existencia de puntos únicos de fallo. Además, al depender de proveedores centralizados, pueden aparecer cuellos de botella que ralentizan la conexión y el acceso a la información (Wan et al., 2023). Estos problemas se abordan en la Web3 que promueve las aplicaciones descentralizadas empoderando al usuario y concibiendo una red más segura y eficiente.

Una aplicación descentralizada (dApp por sus siglas en inglés *Distribuited Application*) no depende de servidores centralizados, funcionan en una red descentralizada de nodos. Las aplicaciones descentralizadas son independientes de administradores o empresas que decidan cómo modificarlas. Aunque es un concepto que existe incluso antes de BC, en la actualidad hay un auge de dApps basadas en BC que simplemente se llaman dApps (P. Zheng et al., 2023) .

Tabla 1. *Stack* de ejemplo para aplicaciones Descentralizadas(Alabdulwahhab, 2018).

Rol	**Web 2.0**	**Web3**
Computación escalable	Amazon EC2,	Ethereum Blockchain
Almacenamiento de archivos	Google Drive	IPFS
Acceso a datos externos	API de terceros	Oráculos
Monetización	Anuncios, venta de productos	Modelo de tokens
Pagos	Paypal	Ethereum, Bitcoin

La creación tanto de una aplicación normal como de una aplicación descentralizada rentable puede requerir computación, almacenamiento de archivos, acceso a datos externos, monetización y pagos. En la Tabla 1 se muestran las tecnologías que se están desarrollado para asumir estos roles en la Web3 en comparación con algunas de ejemplo que se utilizan en la Web 2.0 (Alabdulwahhab, 2018).

En gestión de SLA en redes 5G se pueden aprovechar las tecnologías de la Web3. Para entender cómo, primero es necesario comprender de que se tratan dichas tecnologías, por lo que se exponen brevemente en las siguientes secciones.

2.3.1 Blockchain

Una cadena de bloques (BC, por las siglas del término en inglés Blockchain) es un base de datos distribuida gestionada por una red *peer-to-peer* en la que los registros se almacenan como transacciones en un bloque. Para cada transacción, los datos son firmados y verificados por otros participantes en la red mediante algoritmos criptográficos. Si la mayoría de los participantes está de acuerdo en que la transacción es válida, se añade un nuevo bloque a la cadena de bloques y se comparte con todos los demás nodos (Schweizer Zürich, 2019).

En una BC los "mineros" son actores esenciales pues garantizan la seguridad y el funcionamiento de la red al validar transacciones, crear nuevos bloques y mantener la integridad de la cadena de bloques. Reciben este nombre si la BC utiliza el mecanismo de consenso Prueba de Trabajo (PoW, por las siglas del término en inglés, *Proof of Work*). En el caso de utilizarse otros mecanismos de consenso, como por ejemplo Prueba de Participación (PoS, por las siglas del término en inglés, *Proof of Stake*) los encargados de estas funciones se llaman validadores.

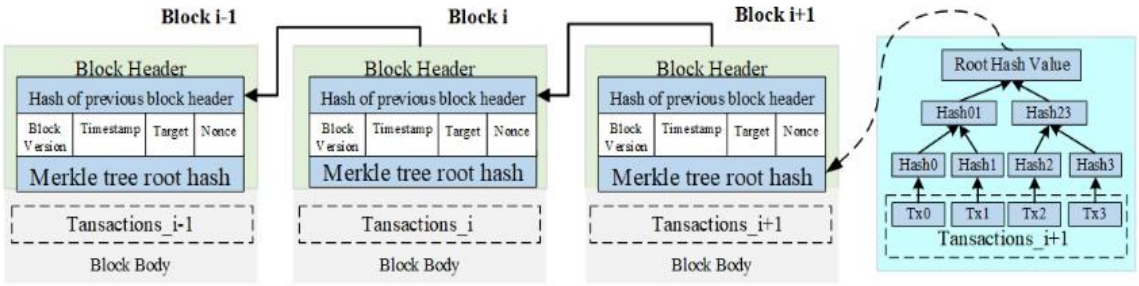

Figura 9. Estructura de datos de Blockchain (S. Zhou et al., 2023).

Los bloques que conforman una BC están compuestos por una cabecera de bloque y un cuerpo de bloque como muestra la Figura 9. El cuerpo del bloque utiliza la estructura de datos del árbol de Merkel para almacenar los datos de la transacción empaquetados por los mineros o validadores, y la cabecera del bloque almacena algunos datos ajenos a la transacción, que se muestran a continuación (S. Zhou et al., 2023):

- Versión del bloque: indica qué conjunto de reglas de validación de bloques seguir;

- El *hash* de la cabecera del bloque anterior: un valor *hash* de 256 bits que se utiliza para implementar enlaces entre bloques apuntando al último bloque;

- *Hash* raíz del árbol de Merkle: registra los valores hash de todas las transacciones del bloque;

- *Timestamp*: se utiliza para registrar la hora de creación de los bloques para garantizar que se crean dentro de un tiempo limitado;

- *Targe*t: objetivo de *hashing* actual en un formato compacto;

- *Nonce*: un campo de 4 bytes que suele empezar por cero y aumenta en cada cálculo *hash*.

Parte importante de las características generales de una BC se debe a esta estructura de datos, que propicia que BC emerja como una innovación revolucionaria, con un potencial significativo para transformar múltiples sectores de la industria. A continuación, se presentan algunas de las características de una BC:

- Descentralización: En los sistemas convencionales de transacciones centralizadas, cada transacción debe ser validada por un organismo central de confianza (por ejemplo, el banco central), lo que inevitablemente genera costos y cuellos de botella en el rendimiento de los servidores centrales. A diferencia del modo centralizado, en BC ya no se necesitan terceros. Se utilizan los algoritmos de consenso en BC para mantener la coherencia y de los datos en la red distribuida.

- Persistencia: Las transacciones son validadas rápidamente y las transacciones inválidas no son admitidas por mineros honestos. Es prácticamente imposible eliminar o deshacer transacciones una vez que se han incluido en la cadena de bloques. Los bloques que contienen transacciones no válidas son descubiertos inmediatamente y no se les permite añadirlas a un nuevo bloque.

- Anonimato. Cada usuario puede interactuar con la BC con una dirección generada, que no revela la identidad real del usuario. Una BC no garantiza la perfecta preservación de la privacidad debido a la restricción intrínseca. (Z. Zheng et al., 2017)

- Auditabilidad: Cada transacción y modificación de datos queda registrada permanentemente y se encuentra disponible para su inspección pública.

Las BC pueden ser permisionadas o no permisionadas. Las BC no permisionadas son plataformas de libros

de contabilidad (*ledgers*) descentralizadas abiertas a cualquiera que publique bloques, sin necesidad de permiso de ninguna autoridad (Yaga et al., 2018), mientras que las permisionadas si requieren del permiso de una autoridad. Las BC públicas se categorizan como permisionadas, y las BC privadas, de consorcio e híbridas como no permisionadas. Las BC públicas se categorizan como permisionadas, y las BC privadas, de consorcio e híbridas como no permisionadas. De forma específica existen las BC públicas, las BC privadas, las BC de Consorcio y las BC Híbridas como se muestra en la Figura 10, sin embargo, las más utilizadas son las BC públicas y las BC privadas.

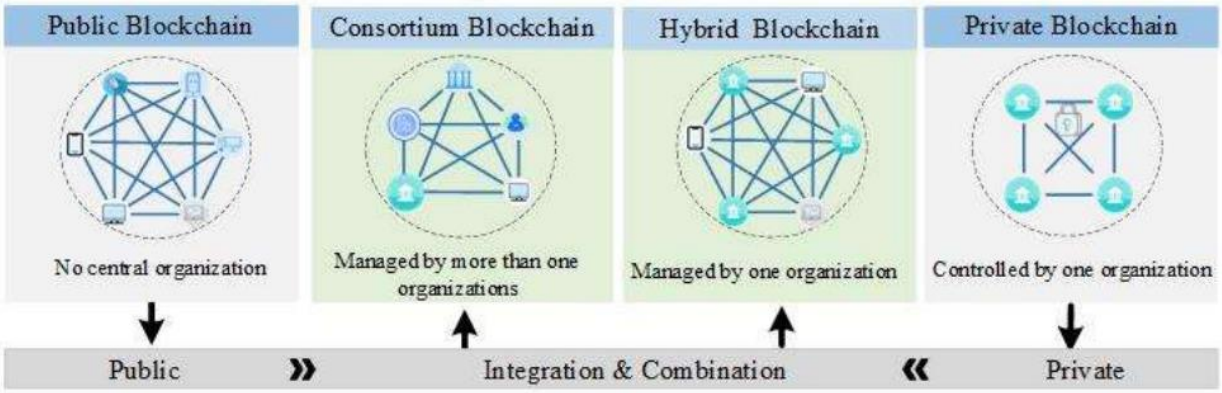

Figura 10. Los cuatro tipos de Blockchain (S. Zhou et al., 2023) .

En una BC pública, todos los registros son visibles para el público y todo el mundo puede participar en el proceso de consenso. Por el contrario, en una BC de consorcio, sólo un grupo de nodos preseleccionados participa en el proceso de consenso. En cuanto a la cadena de bloques privada, sólo los nodos de una organización específica pueden participar en el proceso de consenso. Una cadena de bloques privada se considera una red centralizada, ya que está totalmente controlada por una organización. La BC de consorcio, construida por varias organizaciones, está parcialmente descentralizada, ya que sólo se selecciona una pequeña parte de los nodos para determinar el consenso (Z. Zheng et al., 2017). Las BC híbridas son similares a las BC de consorcio solo que combinan BC privadas y públicas, lo que permite a los usuarios integrar sin problemas una BC privada con varias públicas. A diferencia de las BC de consorcio con múltiples participantes que ayudan colectivamente a mantener la red, una BC híbrida puede tener un administrador de red de una sola entidad. Las BC híbridas funcionan mejor cuando una organización necesita que algunos de sus datos sean de libre acceso y otros se mantengan privados para uso interno (S. Zhou et al., 2023).

La tecnología BC, que inicialmente se desarrolló para abordar la necesidad de una fuente única de confianza en el sector bancario, ahora se está estudiando para brindar la confianza distribuida necesaria

en los ecosistemas de telecomunicaciones en constante evolución. Los nuevos modelos de propiedad de la infraestructura de telecomunicaciones están cambiando de un modelo dominado por unos pocos operadores de red a uno con muchos actores pequeños y grandes. En este nuevo contexto, las fuentes tradicionales de confianza, como las agencias gubernamentales, se vuelven menos pertinentes y se requieren fuentes de confianza distribuidas. Esto se debe a las limitaciones inherentes de los sistemas centralizados, que permiten una escalabilidad limitada y exponen a todo el ecosistema a problemas de punto único de falla (Afraz et al., 2023).

2.3.1.1 Mecanismos de consenso

Adicional a la criptografía, que incluye el hash criptográfico y la firma digital, el éxito clave de Blockchain se basa en los mecanismos de consenso que determinan el rendimiento general y la escalabilidad de un sistema. El consenso se describe como el acuerdo entre un grupo de nodos sobre la verdad de sus datos (Viriyasitavat & Hoonsopon, 2019). Un mecanismo de consenso determina en una BC qué nodo será elegido para la creación de un nuevo bloque y cómo el resto de los participantes de la BC valida dicho bloque. La selección adecuada es de suma importancia, ya que impacta directamente en el desempeño de la BC que lo utilice.

Se utilizan diferentes mecanismos de consenso dependiendo de si es permisionada o no permisionada, y esto se debe a las características intrínsecas que las definen. Para el caso de las no permisionadas, dado que no se conoce la identidad de los usuarios, y estos son muchos, se pueden requerir mecanismos que consuman recursos de cómputo intensivos, pues no existe confianza entre los participantes y algunos pudieran tener intenciones maliciosas.

Se explican brevemente algunos mecanismos de consenso para BC no permisionadas:

- PoW: Si los nodos quieren generar y escribir un nuevo bloque en la BC, deben resolver un complejo rompecabezas fácil de verificar, pero difícil de encontrar. El nodo que primero resuelva la respuesta obtiene los derechos de contabilidad. Después, el nodo que obtiene los derechos contables difunde la respuesta resuelta y el registro de la transacción a otros nodos para su verificación y comienza la siguiente ronda de minería. Si otros nodos participantes validan la transacción del bloque y la respuesta al enigma es correcta, significa que la respuesta es creíble.

El nuevo bloque se escribe en la BC del verificador, y éste entra en la siguiente ronda de minería competitiva (S. Zhou et al., 2023).

- PoS: resuelve los problemas de tiempo y consumo de electricidad que tiene PoW, ya que el requisito de electricidad está asociado a que los mineros encuentren un *nonce* (respuesta al rompecabezas) y este proceso necesita cierto tiempo. PoS hace que los nodos pongan una apuesta para que uno sea elegido como creador del siguiente bloque. Cuando se elige un bloque, el creador recibirá las comisiones de transacción asociadas a ese bloque. Si el ganador de un bloque intenta añadir un bloque no válido, perderá su apuesta (Guo & Yu, 2022).

- DPoS: todos los poseedores de *tokens* pueden votar por un número de delegados y también pueden delegar en otros usuarios con su poder de voto. Cuantos más *tokens* tenga el poseedor más poder tiene el poseedor del *token*. Después, los delegados se encargan de validar las transacciones y los bloques para asegurar la red. A diferencia de la mayor potencia de cálculo en PoW o la mayor cantidad de tokens en PoS, los poseedores de tokens en DPoS pueden votar sobre quién mina los nuevos bloques y recompensar sólo a los mejores mineros (Guo & Yu, 2022).

En el caso de las BC permisionadas la identidad de los participantes es conocida y requieren ser autorizados por lo que los mecanismos de consenso presentan mayor eficiencia, al no tener que lidiar con intenciones maliciosas entre nodos. Estos mecanismos también se deben adaptar para alinearse con requisitos específicos de entornos empresariales. Algunos ejemplos de mecanismos de consenso explicados de forma resumida para BC permisionadas son:

- PoA: Los validadores deben pasar un riguroso procedimiento de investigación antes de que se les permita crear nuevos bloques (Fahim et al., 2023). La esencia del mecanismo de reputación es la confianza en la identidad del validador. Las personas con PoA tienen la motivación de mantener la posición que han alcanzado, ya que deben ganarse el privilegio de ser validadores. Al dar reputación a las identidades, se anima a los validadores a defender el proceso de transacción, ya que no quieren que sus identidades se asocien a una mala reputación (Fahim et al., 2023).

- PBFT: alcanza consenso entre un conjunto conocido de nodos validadores a través de una secuencia de pasos de comunicación, que incluye la elección de un líder primario quien propone nuevos bloques, la propagación del bloque a los demás nodos, la firma del bloque por la mayoría, y la adición del bloque firmado a la cadena. PBFT garantiza consenso mientras no más de un tercio

de los nodos fallen bizantinamente (Z. Zheng et al., 2017).

- Paxos: El mecanismo Paxos incluye tres roles: proponentes, aceptantes y aprendices. El papel del proponente es proporcionar una propuesta o solicitud. La información de la propuesta incluye un identificador y el valor de esta. La función de los aceptantes es participar en la toma de decisiones y responder a las propuestas de los proponentes. Si la mayoría de los aceptantes asienten la propuesta, se dice que ha sido aprobada. Los aprendices no participan en la toma de decisiones, ya que se enteran de la última propuesta acordada por los proponentes o los aceptantes. Aquí un nodo puede desempeñar varios roles (S. Zhou et al., 2023).

- Raft: Posee 3 roles. El estado inicial de todos los nodos es "seguidor". Estado intermedio de transición de seguidor a "líder" es "candidato". El líder es responsable de interactuar con el cliente y de la replicación de *logs* (la replicación de *logs* es unidireccional, es decir, la dirección que el líder envía a los Seguidores). Sólo existe un líder en todo el sistema al mismo tiempo (Hu & Liu, 2020). En el caso de una BC los *logs* replicados sería el nuevo bloque para conformar. Raft se basa en un modelo de liderazgo. En un clúster Raft, un nodo líder es responsable de coordinar y tomar decisiones en nombre del clúster. Si el líder falla, se inicia un proceso de elección de un nuevo líder. A diferencia de Paxos que no se basa en un modelo de liderazgo, todos los nodos participan activamente en la toma de decisiones.

- HoneyBadgerBFT: Es un algoritmo de consenso para entornos asíncronos. La mayoría de los algoritmos de consenso se basan en redes síncronas y eligen a un líder. Es especialmente adecuado para su aplicación en BC, ya que progresa incluso en condiciones muy adversas (Strehl, 2018).

Es importante destacar que la elección adecuada de la BC para desarrollar una dApp requiere evaluar cuidadosamente factores como la escalabilidad para procesar un alto volumen de transacciones de manera eficiente, los costos asociados a dichas transacciones, las características y funcionalidades específicas que ofrece cada BC alineadas con los requisitos del proyecto.

Con el fin de garantizar que la opción seleccionada sea la más adecuada para optimizar el rendimiento, costos y alineación con los objetivos del caso de uso específico, se requiere de la madurez del ecosistema de la BC para aprovechar su soporte y adopción, el modelo de gobernanza y descentralización preferidos, así como los requisitos de interoperabilidad en caso de que la dApp necesite interactuar con otras aplicaciones o BC.

2.3.2 Contratos inteligentes (*Smart Contract*)

Los contratos inteligentes son códigos ejecutables que se corren sobre la BC para viabilizar y dar cumplimiento a un acuerdo entre partes no fiables de forma programática, donde no existe participación de un tercero de confianza (Khan et al., 2021). Se puede considerar un contrato inteligente como una clase que puede contener variables de estado, funciones, modificadores de funciones, eventos y estructuras (Buterin, 2014). En comparación con los contratos tradicionales, los contratos inteligentes permiten a los usuarios codificar sus acuerdos y relaciones de confianza proporcionando transacciones automatizadas sin la supervisión de una autoridad central. Una transacción en este entorno es una operación enviada a la BC que puede incluir la ejecución de funciones dentro de un contrato inteligente, la transferencia de criptomonedas entre cuentas, o la implementación de un nuevo contrato inteligente en la red. Para evitar la manipulación de los contratos, los contratos inteligentes se copian en cada nodo de la red BC. Al permitir la ejecución de las operaciones por computadoras y servicios proporcionados por plataformas BC, se podría reducir el error humano para evitar disputas en relación con dichos contratos (Khan et al., 2021).

Un desarrollador puede escribir un SC y luego desplegarlo en una BC. Luego, este contrato residirá en una dirección específica de la BC y puede ser invocado por cualquier usuario o contrato mediante transacciones. Un contrato inteligente contiene condiciones y lógica predefinidas. Por ejemplo, puede especificar que una cierta cantidad de *tokens* debe ser transferida a un usuario si se cumplen ciertas condiciones (P. Zheng et al., 2023). De forma general se puede decir que un *token* es un objeto físico o digital que puede ser poseído y transferido, que representa algo más y que existe dentro de un sistema (Schwiderowski et al., 2024). En el contexto de BC es un objeto digital que representa ciertos derechos y valores. Un *token* intercambiable que funciona y existe desde la creación de una BC con objetivo comercial, es considerado una criptomoneda.

En un SC una vez que se recibe una transacción que activa una función del contrato, el contrato se ejecuta automáticamente según la lógica definida. Esto puede incluir la transferencia de fondos, el cambio de estados internos del contrato, o la emisión de eventos. La ejecución del contrato es validada por la red de nodos de la BC. Una vez validada, la transacción se registra en la BC, asegurando que es inmutable y verificable (Z. Zheng et al., 2020).

Los SC tienen aplicaciones transformadoras en múltiples áreas. En el sector financiero y de seguros, permiten implementar acuerdos complejos sin intermediarios, automatizando pagos de intereses, reembolsos de préstamos, liquidación de derivados y compensaciones de pólizas (Nzuva, 2019). En el

ámbito de las propiedades, posibilitan crear tokens de activos físicos vinculándolos a documentos legales. En cadenas de suministro y logística, rastrean el movimiento de mercancías, liberan pagos al cumplir hitos y aplican penalizaciones por retrasos o incumplimientos. En el ámbito de identidad y credenciales digitales, proporcionan un marco descentralizado para emitir y verificar certificados educativos, licencias profesionales y otros documentos. En procesos gubernamentales y democráticos, los SC en BC públicas garantizan votaciones y procedimientos transparentes e inmunes a la censura (Lin et al., 2022). Esta tecnología disruptiva basada está sentando las bases para nuevos modelos de negocio y automatización de procesos en prácticamente cualquier industria.

Un concepto muy importante y relacionado con SC es *gas*. El *gas* en BC es una unidad de medida que representa el costo computacional de ejecutar operaciones en la red. Se utiliza principalmente en redes BC para determinar cuánto deben pagar los usuarios por cada transacción o ejecución de una función de un contrato inteligente. El precio del *gas* fluctúa según la congestión de la red y la cantidad de operaciones realizadas durante la transacción (Meister et al., 2024).

Para mejorar la gestión de Acuerdos de Nivel de Servicio en NS los SC ofrecen oportunidades interesantes. Estos contratos inteligentes pueden ser utilizados para codificar y almacenar los términos y condiciones de los SLA de manera inmutable y transparente, garantizando que ambas partes, cliente y proveedor, tengan acceso a la misma información y no puedan modificar los términos de manera unilateral. Los contratos inteligentes pueden ser programados para monitorear automáticamente el cumplimiento de los SLA en tiempo real, integrándose con oráculos, que son sistemas distribuidos que pueden tener acceso a información externa a la BC (Beniiche, 2020) y entregarla a los contratos inteligentes. Si se incumple un SLA, el contrato inteligente activa acciones predefinidas, como multas o compensaciones. En este sentido, se puede explotar la capacidad de los contratos inteligentes para realizar pagos automáticos basados en el cumplimiento de los SLA, de esta forma se reduce la fricción y los retrasos en los procesos de facturación y pago, al tiempo que aumenta la transparencia y la confianza entre las partes y se reduce la carga operativa de los proveedores.

Al registrar todos los eventos y datos relacionados con los SLA en una BC de forma inmutable, se crea un registro auditable y rastreable, se facilita la resolución de disputas y se mejora la responsabilidad de las partes involucradas. La estandarización de los SLA en contratos inteligentes, se puede mejorar la interoperabilidad entre diferentes proveedores de servicios y redes, y facilita la portabilidad de los SLA, así como, la migración de servicios entre diferentes entornos de red.

2.3.3 Monedero digital

Es importante destacar que, sin un monedero de criptomonedas no es posible realizar operaciones en las redes BC. Un monedero digital es un programa diseñado específicamente para interactuar con los datos de una o varias BC y realizar operaciones relacionadas con las criptomonedas o tokens, como visualizar saldos y ejecutar transacciones (Suratkar et al., 2020). Los monederos digitales trabajan con direcciones públicas que son cadenas de caracteres hexadecimales que constituyen identificadores únicos y precisos que se utilizan para recibir un tipo particular de criptomoneda y pueden ser compartidas abiertamente. El monedero brinda acceso a todas las transacciones asociadas con esa dirección en la BC respectiva y permite visualizar los saldos vinculados a una dirección, así como, mover fondos dentro de la BC, siempre que se sea el propietario de dicha dirección (Gentilal et al., 2017).

Una clave secreta está asociada a cada dirección pública de un monedero digital. Esta es difícil de recordar para los seres humanos, por lo tanto, se almacena en el software del monedero o con proveedores de terceros en el caso de monederos en línea. Se puede entender un monedero como la plataforma de tu cuenta bancaria en línea, tu dirección pública es similar a tu número de cuenta, la BC es como el libro mayor del banco, y los monederos son como un custodio (Suratkar et al., 2020). En resumen, el monedero interactúa con la BC utilizando la dirección pública almacenada por medio de transacciones que se firman con la clave privada asociada a la dirección pública.

2.3.4 Oráculos

Los SC tienen una limitante fundamental, no pueden interactuar de forma inherente con datos y sistemas que existen fuera de su entorno nativo. Las BC están intencionalmente aisladas y de ahí se derivan sus propiedades más valiosas, entre las que se encuentran un fuerte consenso sobre la validez de las transacciones de los usuarios, la protección frente a ataques de doble gasto y la mitigación de interrupciones en la red. Para conectar ambos entornos, y lograr una interoperabilidad segura con sistemas externos, una BC demanda una pieza adicional de infraestructura conocida como "oráculo"(Chainlink, 2024b).

Los oráculos sirven como puentes entre las cadenas de bloques y el mundo exterior y esto se debe a que como la BC es un sistema cerrado y determinista, se encuentra aislado y para la ejecución de un contrato real de forma programática se necesitan datos del “mundo real” (Damjan, 2018). Es importante destacar

que el principal uso de la BC es el procesamiento de transacciones de activos digitales, donde los datos requeridos provienen del interior de la propia cadena. No obstante, existen aplicaciones como en cadena de suministros y en Internet de las Cosas que necesitan obtener datos del mundo real para su funcionamiento, y los SC no están diseñados para realizar solicitudes externas y acceder a esta información (Lin et al., 2022). En este sentido, los oráculos juegan un papel fundamental al permitir entregar información fuera de la cadena de forma confiable.

Los oráculos permiten a SC tener acceso a varios tipos de datos fuera del dominio de BC, por ejemplo, temperaturas, resultados deportivos, confirmación de realización de un pago o cualquier tipo de información, provenientes de fuentes como una API, una página Web o una aplicación segura que se efectúe en un entorno de ejecución confiable (TEE, por las siglas del término en inglés, *Trust Execution Enviroment*) (Beniiche, 2020). Luego de extraer los datos los procesan, lo que incluye verificación y validación, y los formatean. Posteriormente se firman digitalmente aplicando criptografía de clave pública/clave privada, lo que asegura que los datos provienen de una fuente legítima y no han sido alterados. Una vez firmados están listos para transmitirse a la BC y esto es posible ya que el oráculo tendrá un nodo conectado a la red.

La Figura 11 muestra cómo opera un oráculo en una BC. Se observa cómo un SC envía una solicitud al contrato inteligente del oráculo. Este último, gracias a su función de automatización, realiza el mecanismo correspondiente del oráculo según la solicitud recibida para interactuar con una API externa. Si se utiliza un único mecanismo de oráculo, el sistema externo puede transferir datos directamente a la BC. Sin embargo, si se ejecutan múltiples mecanismos de oráculos distribuidos, es necesario utilizar el consenso de varios nodos oráculos participantes para alcanzar los resultados finales y transferirlos desde o hacia la API externa, logrando así una interacción de datos descentralizada. Una vez que el oráculo de la BC automatizado se despliega con éxito, se comunica con la fuente de datos externa para recopilar los datos requeridos. De esta forma, el solicitante puede visualizar esos datos al ejecutar su propio contrato inteligente (Beniiche, 2020).

La utilización de oráculos distribuidos aumenta la confiablidad y disponibilidad de los datos externos extraídos del exterior en comparación con los oráculos centralizados. En trabajos como (De Brito Gonçalves et al., 2020) se plantea la utilización de oráculos distribuidos para aumentar la confiabilidad en las métricas de monitoreo.

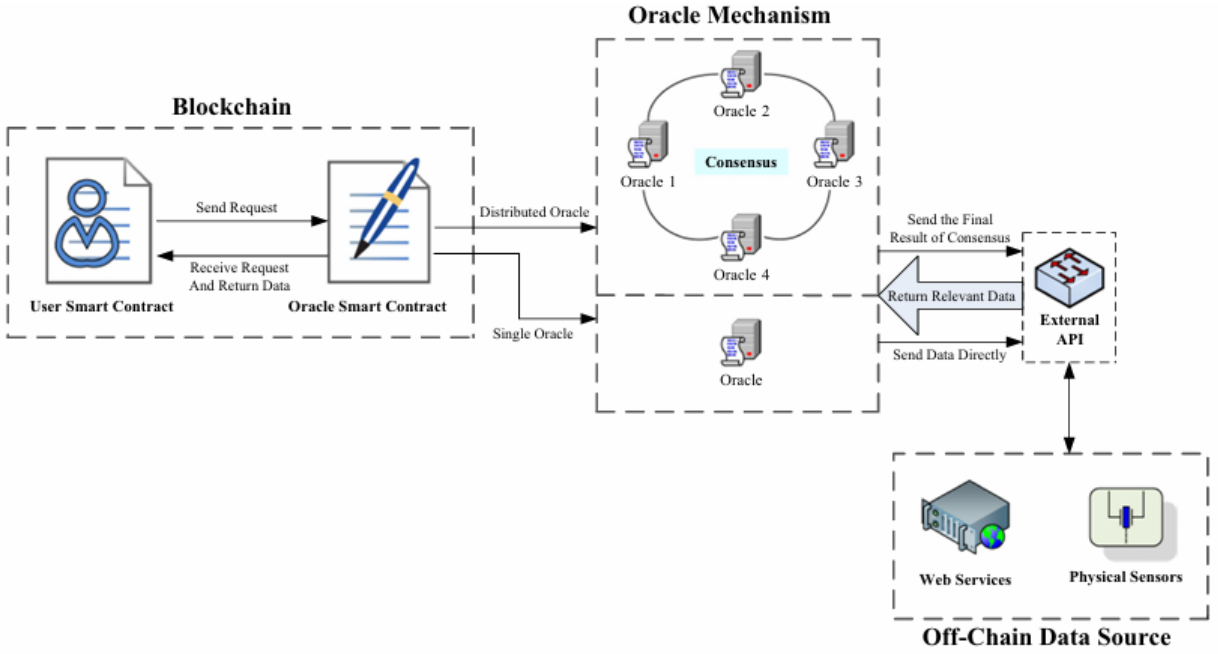

Figura 11. Operación de un Oráculo en Blockchain (Beniiche, 2020).

2.3.5 Sistema de archivos interplanetario IPFS

El sistema de archivos interplanetario (IPFS) es un sistema de archivos distribuido *peer-to-peer* que busca conectar todos los dispositivos informáticos con el mismo sistema de archivos. En otras palabras, IPFS proporciona un modelo de almacenamiento en bloques de alto rendimiento encaminado a contenido, con hipervínculos dirigidos precisamente a contenido (Benet, 2014).

La red IPFS está formada por computadoras que ejecutan el software cliente IPFS. Cualquier persona puede unirse a esta red, ya sea ejecutando un nodo IPFS o simplemente utilizándola como usuario para almacenar y recuperar archivos. Se pueden guardar todo tipo de archivos como texto, música, videos e imágenes (N. Sangeeta & Nam, 2023).

En IPFS hay tres procesos fundamentales (Rivas & Mackarena, 2022):

- La identificación única por medio de direccionamiento de contenido: a diferencia de HTTP, en IPFS los datos se identifican por su contenido no por su ubicación. Al subir un archivo a IPFS, se genera un *hash* único del contenido que lo identifica. Si se sube un archivo diferente, su *hash* será completamente distinto, y se puede recalcular localmente para verificar que coincida con el original de IPFS (N. Sangeeta & Nam, 2023).

- La vinculación de contenido a través de Grafos Acíclicos Dirigidos (DAG, por las siglas del término en inglés, *Directed Acyclic Graph*): Un DAG es una estructura de datos en la que los nodos están conectados direccionalmente y no forma bucles, como se observa en la Figura 12. En el contexto de IPFS, se utiliza para representar la estructura de datos de los objetos almacenados en la red de esta forma cada nodo del grafo contiene una porción de datos y enlaces a otros nodos que representan partes relacionadas del mismo objeto. Esto permite que los archivos se dividan en fragmentos más pequeños, con cada fragmento hace referencia a los otros necesarios para reconstruir el archivo completo. La utilidad principal de utilizar un DAG en IPFS es que permite una mayor eficiencia en el almacenamiento y la recuperación de archivos pues dividen en bloques más pequeños, lo que facilita la distribución y la redundancia en la red. Que no haya ciclos en la estructura de datos simplifica la lógica de búsqueda y evita problemas de referencia circular (Benet, 2014).

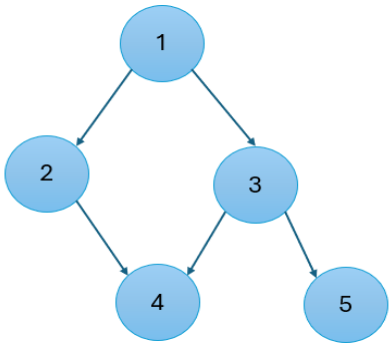

Figura 12. Ejemplo de DAG [Elaboración propia].

- El descubrimiento de tablas de hash distribuidas (DHT, por las siglas del término en inglés, Distributed Hash Table): DHT es un sistema descentralizado para almacenar y recuperar datos en una red peer-to-peer. En una DHT, los datos se almacenan en varios nodos de la red en lugar de en un servidor centralizado. Cada nodo en la red tiene una porción de la tabla de hash, que se utiliza para mapear claves a valores. Cuando un nodo necesita recuperar un dato, calcula el hash de la clave asociada al dato. Luego, utiliza la tabla de hash distribuida para encontrar los nodos responsables de esa clave y solicita el dato a esos nodos (Rivas & Mackarena, 2022).

Un concepto muy relacionado con esta tecnología es el concepto de *hash* criptográfico. Este es una función matemática que toma una entrada y genera una salida de longitud fija, generalmente en forma de una cadena de caracteres. Esta salida, también llamada hash, es única para cada entrada distinta y es prácticamente imposible de revertir para obtener la entrada original. Se utiliza para verificar la integridad

de los datos y asegurar la autenticidad de la información.

BC tiene limitaciones en cuanto su almacenamiento pues, al tener una estructura de datos inmutable, se ocupa más espacio con cada nueva transacción. Por esta razón no es viable guardar archivos o gran volumen de datos directamente en una BC. Una solución es conservar el archivo o los datos en IPFS y solamente guardar en BC el *hash* criptográfico vinculado que devuelve la red IPFS al subir un archivo. De esta manera se combina la confiabilidad y persistencia de datos de BC con la disponibilidad, integridad y eficiencia que otorga IPFS. Dada la naturaleza distribuida de ambas tecnologías se pueden considerar aliados naturales (F. Yang et al., 2023).

2.4 BC y SC en la arquitectura SBA:

Los avances en la tecnología de redes 5G han generado una necesidad imperante de gestionar y optimizar los recursos de red de forma eficiente con el fin de garantizar una QoE satisfactoria. A medida que aumenta la complejidad de la red y la demanda de QoS alta, la gestión de redes se ha vuelto cada vez más desafiante (Gonzalez-Franco et al., 2023). En este contexto, se requieren enfoques nuevos y soluciones innovadoras para abordar los retos emergentes en la gestión y monitoreo de recursos de red de manera automatizada y mantener altos niveles de confiabilidad, eficiencia y seguridad en la relación cliente-proveedor. En este sentido, los SLA entre proveedores y clientes verticales (u otros proveedores), posibilitan un marco de entendimiento en la negociación de NS en las redes 5G. Sin embargo, estos acuerdos necesitan tener transparencia, confianza y automatización. La información del desempeño de los servicios prestados debe ser auditable y estar dotada de la trazabilidad necesaria para que los clientes tengan la certeza del cumplimiento o no del SLA contratado, y se realicen las compensaciones pertinentes por violaciones o incidencias en la prestación del servicio. Para abordar esta problemática la utilización de BC y SC se erigen como una solución prometedora.

Blockchain promueve un entorno donde no se necesita confianza entre los participantes de la red. En redes heterogéneas con múltiples operadores conectados, como la red 5G, tal entorno podría ser ventajoso. Un conjunto de servicios, tal vez implementados como contratos inteligentes, puede estar disponible para todos los interesados, sin necesidad de que los operadores arbitren o cedan responsabilidades (Nguyen et al., 2021).

Otro elemento que alienta la utilización de BC en redes de nueva generación, es que está diseñada para ser altamente resistente a la manipulación, lo que potencialmente puede mejorar la integridad y confiabilidad de los datos relativos al monitoreo de SLA (Javed & Mangues-Bafalluy, 2023). Gracias a las cualidades de inmutabilidad de BC, al utilizarla para almacenar la información del monitoreo, los datos obtenidos del comportamiento de un NS son difícilmente corrompibles.

Con la llegada de 5G, el modelo tradicional de negocio de las redes móviles está cambiando de un modelo centrado en los operadores de red, a un sistema más abierto con varios actores. En este escenario, el proveedor de NS actúa como intermediario entre el proveedor de recursos de infraestructura y el proveedor de servicios vertical.

El proveedor del NS necesita entonces un mecanismo de intermediación que le permita arrendar recursos de varios proveedores de forma segura y privada para implementar un NS. Con esta idea trabajos como (Nour et al., 2019) involucran a las tecnologías BC y SC como herramientas para construir un mecanismo de intermediación en el arrendamiento de segmentos de red demostrando que es viable, de forma que se eliminan intermediarios en el proceso.

Evitar la burocracia, dependencia y costos asociados causados por la intervención de un tercero es posible con la utilización de la dupla BC y SC. No solo con respeto a suprimir un mediador al ocurrir discrepancias en la verificación de la entrega de QoS establecida en el SLA, sino también en la dependencia de elementos externos para la realización de pagos. Con SC se puede gestionar de manera automática el proceso de facturación del SLA, que comprende el pago del servicio por parte del cliente y el reembolso de la compensación por parte del proveedor (Scheid et al., 2019).

La automatización mencionada anteriormente es muy importante ya que a pesar del uso de Sistemas de Soporte Operativo (OSS, por las siglas del término en inglés, Operative Support System) y Sistemas de Soporte Empresarial (BSS, por las siglas del término en inglés, Business Support System) en el establecimiento SLA entre un cliente y un proveedor, ciertas tareas como la especificación de los SLA y el cálculo de compensaciones siguen requiriendo un esfuerzo manual y una interacción directa entre las partes. Esta necesidad de intervención humana dificulta la agilidad en la prestación de servicios y propicia la aparición de errores (Scheid & Stiller, 2018).

Luego de un estudio de artículos que han abordado el uso de BC y SC para la gestión de SLA se extrajeron características relevantes de cada uno. En la tabla 2 se comparan los diferentes artículos analizados.

Tabla 2. Comparación entre sistemas de gestión de SLA que usan BC y SC [Elaboración Propia].

Artículo	Monitoreo	Intermediación en la negociación	Blockchain (Mecanismo de consenso)	Comparación de implementación en diferentes BC	Análisis de desempeño	Estudio e influencia de Parámetros de QoS
The Confluence of Blockchain and 6G Network: Scenarios Analysis and Performance Assessment (Li et al., 2022)	No	No	Quorum (IBFT)	No	Si	No
Service-Level Agreement Management with Blockchain-based Smart Contract to Improve the Quality of IT Service Management (Azzahra & Nugraha, 2023)	Si	Si	BC Privada (POA)	No	Si	Disponibilidad de Incidentes de Servicio, Disponibilidad de Solicitudes de Características del Servicio, Disponibilidad de Solicitudes de Datos del Servicio, Tiempo de Soporte
A Trust architecture for the SLA management in 5G networks (Saad et al., 2021)	Si	No	Ethereum (POW)	No	No	Latencia, througput
A Quality-of-Service Compliance System Empowered by Smart Contracts and Oracles (De Brito Gonçalves et al., 2020)	Si	No	Ethereum (POW)	No	Si	No
Blockchain and Smart Contracts for Telecommunications: Requirements vs. Cost Analysis (Afraz et al., 2023)	No	Si	Hyperledger Fabric (Sin especificar)	No	Si	No

Para la conformación de la tabla se utilizaron criterios comparativos como la existencia de capacidad de monitoreo, la integración de funciones de intermediación en la negociación, el tipo de BC utilizada, la realización de comparaciones de la implementación en diferentes BC y el estudio e influencia de los parámetros de QoS que se tienen en cuenta para el monitoreo.

2.5 Conclusiones

En este capítulo se destacó la transición de las redes 4G a 5G y la importancia de la virtualización de funciones de red para una mayor flexibilidad y eficiencia. Se subrayó la tecnología de NS en 5G, que permite crear redes virtuales optimizadas para aplicaciones específicas. Se discutió la relevancia de los Acuerdos de Nivel de Servicio (SLA) y los desafíos de su gestión automatizada, asimismo, se presentó el potencial de tecnologías disruptivas como BC, SC y otras para mejorar la gestión de SLA en redes 5G/B5G. A partir de lo expuesto se concluye que:

- La existencia actual de diversidad de servicios y sus requisitos impulsan un enfoque diferente a la arquitectura rígida de 4G. Por esta razón hay una evolución a una arquitectura basada en servicios más flexible, donde NS constituye un papel fundamental.

- NS abre la posibilidad a nuevos modelos de negocio donde interesados verticales pueden arrendar segmentos de red para satisfacer sus requisitos de comunicación.

- La NEF constituye un punto de conexión entre el núcleo de red de 5G y aplicaciones de terceros haciendo posible la integración de las capacidades de la red con aplicaciones externas para el monitoreo de NS.

- Los SLA constituyen un marco para el entendimiento entre proveedores y clientes pues definen las responsabilidades de cada una de las partes y tiene un ciclo de vida que puede ser automatizado.

- La red IPFS posibilita el almacenamiento distribuido de archivos aumentando su disponibilidad y garantiza la integridad de los datos almacenados. Su combinación con BC es una opción natural pues complementa sus limitaciones en escalabilidad.

- BC y SC son tecnologías disruptivas cuyas características puede ser aprovechadas para la intermediación en la negociación de NS, la monitorización del cumplimiento del SLA asociado y la realización programática de pagos. Constituyen tecnologías prometedoras en la automatización del ciclo de vida de SLA de NS.

Capítulo 3. Propuesta de arquitectura para gestión de SLA de NS

En el contexto de las redes 5G y más allá, NS brinda la capacidad de virtualizar los recursos de red para satisfacer los diversos requisitos de diferentes casos de uso. No obstante, la gestión de SLA para estos NS sigue representando un desafío debido a la naturaleza dinámica y compleja de los entornos de red. En este sentido, se puede aprovechar la tecnología de BC y SC para facilitar una gestión transparente, confiable y automatizada de los SLA de NS. Los SC, al ser piezas de código ejecutables en un entorno BC, pueden garantizar la ejecución confiable de acuerdos entre dos partes no confiables, como es el caso de un SLA. Es factible llevar a cabo cada una de las fases del ciclo de vida de un SLA al utilizar esta tecnología, desde la definición del acuerdo hasta su terminación.

En este capítulo, se propone un esquema de gestión para los SLA de NS y se detallan cada uno de sus componentes, justificando su inclusión. Entre estos componentes se encuentran una dApp, la red IPFS, SC, un oráculo distribuido y una API para el acceso a datos de monitoreo de NS. Con su integración, se pretende proporcionar un diseño de sistema eficiente, seguro, confiable, transparente y automatizado. En las secciones siguientes, se plantean las interacciones entre los componentes como procedimientos que hacen cumplir el ciclo de vida de un SLA. Al finalizar, se realizan recomendaciones respecto a la implementación del esquema, abordando criterios de elección y comparando las tecnologías a utilizar en algunos de los componentes.

3.1 Propuesta de arquitectura para gestión de SLA de NS aplicando BC y SC

Tal como se indicó en el Capítulo 1 y § 1.1, en la gestión de los SLA de NS de redes de nueva generación existen problemas de confiabilidad por parte del cliente ya que el proveedor es quien administra los datos de monitoreo del NS y el cliente no tiene transparencia con respecto a la entrega de los parámetros de QoS plasmados en el SLA. El proveedor puede actuar de forma maliciosa sin notificar de violaciones en el SLA viéndose el cliente afectado. Por otro lado, el proveedor tiene que lidiar con intensiva carga de trabajo al tener que establecer acuerdos, analizar reportes de monitorización, gestionar cobros y compensaciones y dar mantenimiento a la infraestructura tanto para gestionar los NS como la gestión de los SLA. A los problemas anteriores se suma el inconveniente que representa la intervención de un intermediario en caso de disputas o discrepancias entre cliente y proveedor. Un intermediario o mediador introduce costos adicionales, demoras, burocracia y entorpece la gestión de SLA.

La propuesta que se muestra en la Figura 13 busca dar solución a estas inquietudes. Se otorga la confianza y transparencia requerida por el cliente con respecto a la QoS entregada al incorporar el uso de BC para almacenar los datos de monitoreo. Los datos relativos al comportamiento del NS son accesibles por el cliente y son prácticamente incorruptibles e inmutables con la utilización de BC. En el esquema se utilizan SC para la automatización de la negociación, monitoreo y gestión de pagos lo que conduce a una reducción del trabajo para el proveedor y se automatiza el proceso de gestión de SLA. El proveedor delega estas funcionalidades en la operación de los SC que funciona en la infraestructura de una BC pública, la cual presenta alta disponibilidad al ser descentralizada y estar compuesta por gran número de nodos. Las operaciones de SC en la gestión de pagos y compensaciones son respuestas a condiciones programadas bajo el consentimiento de proveedor y cliente con el establecimiento de un SLA y evitan la necesidad de un mediador o intermediario con los problemas que representa. Los SC permiten la ejecución de acuerdos sin terceros intermediarios.

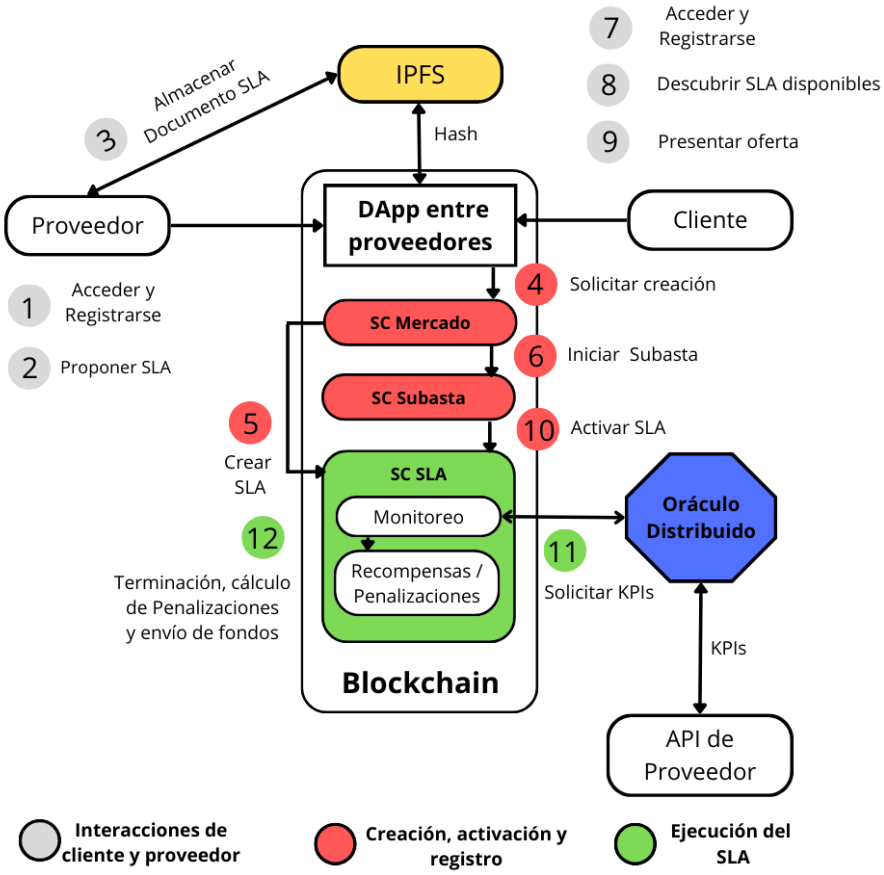

Figura 13. Propuesta de gestión de SLA de NS [Elaboración Propia].

En esta propuesta se integra funciones de monitoreo, intermediación en la negociación y gestión de pagos y recompensas, se considera la utilización de múltiples métricas de monitoreo y sus implicaciones para el

diseño, y se utiliza un oráculo distribuido para atacar la falta de confiabilidad y/o disponibilidad de las métricas de monitoreo. Estos aspectos fueron problemas detectados en la literatura analizada en la sección 1.2 y en esta solución son abordados para mejorar la gestión de SLA de NS al utilizar BC y SC.

En la Figura 13 se muestra el esquema general propuesto para gestionar SLA de NS. En el diagrama se indica a bloques, las partes involucradas, así como la secuencia para la gestión de SLA, la cual está dividida en 3 grupos: las interacciones clientes y proveedor (indicada con el color gris), la creación, activación y registro del SLA (color rojo) y la de ejecución del SLA (color verde).

Este esquema está concebido para ser mantenido por varios proveedores una vez implementado, o sea constituye un entorno de multi-proveedor. En dicho entorno los clientes interesados en arrendar una NS serán proveedores de servicios verticales u otros proveedores de NS. El arrendamiento será posible por medio de una dApp, la cual servirá de interfaz para la interacción de clientes y proveedores con los SC que corren sobre la BC. Para participar en esta interacción ambos actores requieren de monederos digitales con direcciones públicas correspondientes a la BC que se utilice. Cada dirección publica tendrá asociada una clave privada que será utilizada para firmar las transacciones.

En secciones anteriores se explicaba que una transacción pueden ser una transferencia de criptomonedas, despliegue de un contrato inteligente o la ejecución de una función de un contrato inteligente. Por ejemplo, en el caso del registro este será una transacción que consistirá en llamar una función del SC Mercado para guardar un nuevo proveedor, de forma que se le dan capacidades al proveedor para hacer propuestas de NS. Solo un proveedor es capaz de agregar a otro y esto es una regla interna que se puede programar en el SC.

Un proveedor puede registrarse, acceder a la dApp y hacer propuestas de SLA para que estas sean validadas y por la dApp. En esta propuesta se plantean todos los parámetros necesarios para definir el SLA con datos que serán guardados de forma inmutable en la BC. Sin embargo, esta propuesta va acompañada de una petición de almacenamiento del documento SLA correspondiente el cual será conservado en la red IPFS con el fin de evitar consumo de espacio excesivo en la BC. En ella solamente se guarda el *hash* asociado a ese documento que es devuelto por la red IPFS después de confirmar su almacenamiento.

Con los datos relativos a la propuesta de SLA y el *hash* del documento de SLA correspondiente la dApp solicita la creación de un SLA al SC Mercado el cual creará el SC SLA, pero sin un cliente asignado. Con el fin de asignarle un cliente, al tiempo que se crea el SLA, se inicia una subasta creando un SC Subasta. En

este momento un cliente que también tiene posibilidades de registrase y acceder a la dApp, puede enviar ofertas a esta subasta. Una vez terminado el tiempo de la subasta, se determina la oferta ganadora entre los clientes y se activa el SC SLA correspondiente a la propuesta de SLA para realizar el monitoreo del NS contratado. De no haber ofertas se define el SLA como terminado. Es importante destacar el cliente también tiene posibilidad de hacer un descubrimiento de los SLA de NS disponibles por medio de la dApp, estos van a estar almacenados en los registros del SC Mercado.

Al activarse el SC SLA comienza el monitoreo del NS. El contrato inteligente necesita acceder a las métricas de monitorización del NS por lo que hace una solicitud a un oráculo distribuido para que le proporcione esa información proveniente del mundo exterior, que no es más que los KPI respectivos del acuerdo. El oráculo tomará esta información de la API que expone el proveedor correspondiente, API que está concebida a partir de la aplicación de reglas a la NEF del Core de 5G sobre el cual opera este proveedor. En la Figura 14 se puede observar cómo se conecta por medio de esta NF el oráculo al CN de 5G.

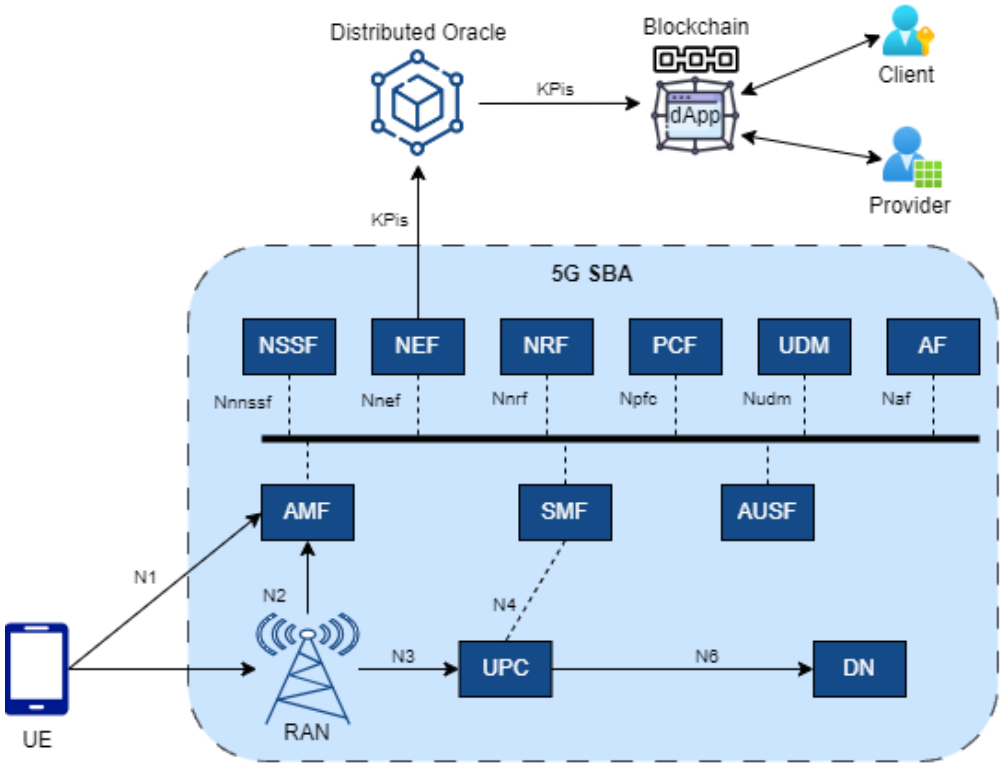

Figura 14. Relación del Core 5G con esquema de gestión de SLA [Elaboración propia].

El SLA tendrá un periodo de operación determinado por las condiciones establecidas en la propuesta de SLA inicial. Durante el periodo, el SC SLA estará comprobando incidencias y violaciones haciendo comparaciones con los umbrales de parámetros de QoS del acuerdo. El número y tipo de violaciones se

acumula hasta la finalización del contrato, y se utiliza esta información en su terminación, donde realiza los cálculos de penalizaciones y hace el envío de fondos respectivos por compensación a los clientes por deficiencias en el servicio prestado si ocurrió alguna, de forma automática y sin una autoridad centralizada. También al finalizar se hace el pago a los proveedores.

Hasta ahora se explicó de forma general la propuesta, en ella se manifiestan las responsabilidades que asume el esquema que son la de intermediación en la negociación de SLA de NS, monitoreo de sus parámetros y gestión de pagos y compensaciones.

3.2 Aplicación distribuida (dApp)

La aplicación distribuida (dApp) propuesta actúa como un puente que permite utilizar la tecnología Blockchain como un *backend* descentralizado, donde se almacena de forma inmutable e incorruptible toda la información relacionada con el establecimiento y gestión de los Acuerdos de Nivel de Servicio (SLA). Esta dApp debe tener una interfaz de usuario (*frontend*) intuitiva y accesible que será utilizada tanto por los clientes como por los proveedores de servicios para llevar a cabo sus respectivas actividades en el ciclo de vida de los SLA. Toda la información necesaria para una interacción con la BC se exigirá en campos a llenar en este *frontend*. El *frontend* de la dApp puede ser desplegado como una aplicación Web tradicional, ya sea en un servidor Web centralizado o de manera descentralizada utilizando servicios como IPFS, Swarm o redes de almacenamiento descentralizado.

El acceso a esta dApp puede ser por medio de un navegador Web mediante una dirección URL si se implementa como una aplicación Web, aunque también podría implementarse una aplicación móvil. Para el caso Web los usuarios, tanto cliente como proveedor, deben instalar una extensión para el navegador que sea un monedero digital, con el fin de posibilitar la interacción con BC. La extensión permite que el usuario firme sus transacciones y pueda acceder a la información de forma segura a la BC.

A diferencia de las aplicaciones Web tradicionales de la Web 2.0, donde las operaciones y la lógica de negocio se ejecutan en servidores Web centralizados, en esta dApp, dichas operaciones serán realizadas por SC desplegados en la BC. Estos contratos inteligentes codifican de manera transparente y auditable las reglas y condiciones de los SLA, así como las acciones a tomar en función del cumplimiento o incumplimiento.

La interacción con los SC no se realiza mediante solicitudes HTTP convencionales, sino a través de llamadas JSON RPC (*Remote Procedure Call*) que es un un protocolo de comunicación remota que se utiliza para la transferencia de datos el formato de datos JSON (*JavaScript Object Notation*). Estas llamadas permiten ejecutar funciones específicas de los contratos inteligentes, y se desencadena así la ejecución de la lógica previamente codificada en ellos. RPC actúa como una interfaz que permite a los usuarios y aplicaciones externas interactuar con nodos de la red de forma remota, este es el caso de la dApp.

Un *endpoint* es un punto final de comunicación en una API que permite el intercambio de datos entre sistemas, generalmente identificado por una URL específica y asociado a una función o recurso particular. Los nodos de la red exponen una serie de *endpoins* RPC, que aceptan solicitudes de usuarios externos y ejecutan operaciones específicas en la BC en nombre del usuario.

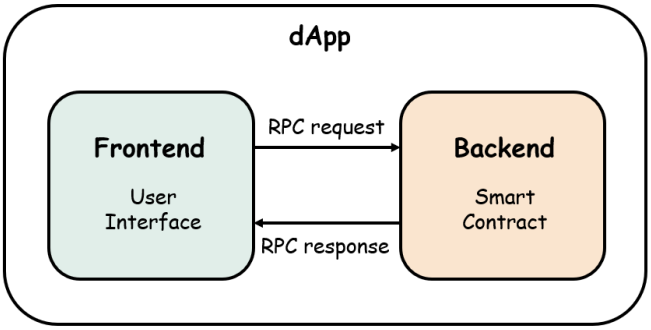

Figura 15. Composición de la dApp [Elaboración propia].

La dApp aprovechará las características inherentes de la tecnología Blockchain, como la transparencia, la inmutabilidad y la trazabilidad, para brindar un entorno seguro y confiable en la gestión de los SLA. Todas las transacciones y eventos relacionados con los acuerdos quedarán registrados de forma permanente e inalterable en la cadena de bloques, lo que permitirá a las partes involucradas auditar y verificar el cumplimiento de los compromisos adquiridos.

3.3 Papel de la red IPFS

Las BC están diseñadas principalmente para almacenar transacciones y datos pequeños, no para almacenar grandes cantidades de datos multimedia o archivos voluminosos. Almacenar archivos directamente en la cadena de bloques haría que su tamaño crezca rápidamente, lo que dificultaría su mantenimiento y propagación a través de la red. Cada nodo integrante de la red debe almacenar una copia completa de la cadena de bloques. Si se almacenan archivos directamente en la BC, esto implicaría que

cada nodo tendría que almacenar una copia redundante de todos los archivos, lo que consumiría una gran cantidad de espacio de almacenamiento innecesariamente.

En la mayoría de las BC públicas se requiere pagar una tarifa de *gas* para almacenar datos en la cadena de bloques. Almacenar archivos grandes directamente en la BC sería extremadamente costoso debido a las altas tarifas de *gas* necesarias. Los datos almacenados no pueden ser modificados una vez que se han agregado a la cadena de bloques. Sin embargo, los archivos a menudo necesitan ser actualizados o reemplazados, lo que haría que el almacenamiento directo en la BC sea ineficiente. Además, acceder a archivos almacenados directamente en la BC puede ser lento debido a la naturaleza distribuida de la red y la necesidad de descargar y verificar toda la cadena de bloques para acceder a los datos.

Por lo anteriormente expuesto se considera una mejor práctica almacenar los archivos en sistemas de almacenamiento descentralizados y distribuidos como IPFS y hacer referencia a ellos desde los contratos inteligentes en la BC utilizando su *hash.* Esto permite aprovechar las ventajas de la BC para almacenar datos críticos y transacciones, mientras que los archivos se almacenan de manera eficiente y escalable en IPFS (F. Yang et al., 2023).

En el caso del esquema propuesto, los clientes pueden requerir tener acceso al documento completo del SLA para analizar en detalle sus términos, condiciones y especificaciones. Para ello, los clientes deben realizar un proceso de descubrimiento de los SLA disponibles, esta información se solicita a la dApp. Como parte de la respuesta a esta solicitud, la dApp proporcionará información relevante sobre los SLA, incluyendo el *hash* asociado al contrato inteligente que codifica los términos del acuerdo. Este *hash* actúa como un identificador único e inmutable, permitiendo a los clientes utilizar la red descentralizada IPFS para descargar el documento oficial y completo del SLA.

La integración con IPFS desempeña un papel crucial en este proceso, brindando una solución de almacenamiento distribuida y altamente disponible para los documentos y archivos relacionados con los SLA. Al almacenar los documentos de los SLA en IPFS en lugar de hacerlo directamente en la BC, se evitan los problemas de escalabilidad, costos elevados y redundancia de datos que conllevaría almacenar archivos voluminosos en la cadena de bloques.

Al utilizar el *hash* proporcionado por la dApp, los clientes pueden tener acceso a la red IPFS y descargar de manera segura y verificable el documento completo del SLA. Gracias a la naturaleza descentralizada de IPFS, los documentos se replican y almacenan en múltiples nodos de la red, lo que garantiza su

disponibilidad y resistencia a fallas o ataques. IPFS proporciona mecanismos de verificación de integridad, lo que permite a los clientes asegurarse de que el documento descargado no ha sido alterado o manipulado, gracias a la inmutabilidad de los *hashes* criptográficos utilizados como identificadores.

De esta manera, la combinación de la dApp basada en contratos inteligentes y la integración con IPFS facilita un acceso transparente, seguro y escalable a los documentos críticos relacionados con los SLA, al tiempo que aprovecha las ventajas de la descentralización y la inmutabilidad proporcionadas por la BC y las redes distribuidas.

3.4 Funcionalidades de los Contratos Inteligentes

En el esquema propuesto (ver Figura 13), se puede apreciar que hay tres SC que se encargan de ejecutar las funciones necesarias para la gestión eficiente de SLA. El primero de ellos es el SC Mercado, el cual actúa como el núcleo central de la dApp, siendo responsable principal de la validación y adición de información a la BC. Este SC contiene el código necesario para el registro de clientes y proveedores, la creación de nuevos SLA y la gestión de subastas para estos acuerdos.

El SC Mercado desempeña un papel crucial, ya que se encarga del lanzamiento de eventos cada vez que se agrega información nueva a la BC. Estos eventos quedan registrados de forma inmutable en la cadena de bloques, y garantiza su integridad, transparencia y trazabilidad. Cualquier participante de la red BC puede tener acceso a los eventos históricos y realizar auditorías o análisis sobre ellos.

Es importante destacar que cuando se despliega un contrato, éste se guarda con una dirección pública, que es un identificador alfanumérico único en la BC, esta dirección se utiliza para interactuar con un contrato específico. El SC Mercado es el primer contrato que se despliega, ya que la dApp basa su funcionamiento principal en este contrato. Actúa como un coordinador de las operaciones principales que pueden realizar los clientes y proveedores, orquestando las interacciones entre los diferentes actores y componentes del sistema.

La propuesta de SLA que hace un proveedor debe venir acompañada de varios parámetros, entre los cuales deben estar: el intervalo de subasta, la dirección del proveedor que define el beneficiario de la subasta y un valor mínimo de ofertas. Estos datos se utilizan para el despliegue del SC Subasta, que se realiza después del despliegue del SC SLA, asimismo se utiliza la dirección de SC SLA como un dato principal en SC Subasta.

Los clientes tienen la opción de descubrir los SLA disponibles por medio de la dApp. Al tener interés en determinado SLA utilizan la dirección de SC Subasta correspondiente proporcionada por la dApp para hacer una oferta a un SLA específico. La oferta debe ser mayor que el valor mínimo y superior a la mayor oferta realizada hasta el momento. Cada oferta va acompañada de la dirección pública del monedero digital de quien la realiza y el valor transferido en la criptomoneda correspondiente a la BC utilizada.

SC subasta tiene una función de retiro que puede ser llamada por los clientes cuya oferta haya sido superada por otra, para recuperar sus criptomonedas. Se debe tener en cuenta que los activos ofertados son custodiados por el SC Subasta. Una vez culmina la subasta, el valor de la mayor oferta se transfiere al SC SLA y se le asigna un cliente al contrato. De no obtenerse ofertas mayores al mínimo establecido por el proveedor, se declara el SC SLA como terminado, queda inactivo y sin cliente asignado.

En la operación del esquema, el SC SLA se crea y despliega por el SC Mercado previo al SC Subasta. Al terminar la subasta con un cliente ganador, se asigna el cliente y comienza a operar el contrato. Su función principal es solicitar información a un oráculo distribuido para que este tenga acceso al “mundo exterior” y obtener los datos relativos al comportamiento del NS contratado. El SC SLA procesará estos datos, asignará el formato adecuado y comprobará la ocurrencia de violaciones o incidencias. Si se alcanza la fecha de expiración del SLA, el SC SLA calcula las penalidades y termina el contrato, se transfiere el pago correspondiente al proveedor y las compensaciones al cliente como resultado de aplicar penalizaciones al proveedor.

El proveedor al finalizar el contrato recibe un pago único al que se aplica una tarifa por transacción, esta tarifa se aplica como pago por la utilización de la BC pública. No se realiza una facturación mensual o por periodos debido a esto requiere realizar varias transacciones y pagar una tarifa por cada una. Esta última variante representa un mayor costo, la opción de realizar un pago único es más eficiente.

3.5 Oráculo distribuido

Para obtener datos del mundo exterior, el esquema plantea utilizar un oráculo distribuido en lugar de uno centralizado. Este enfoque tiene ventajas relevantes. Un oráculo distribuido está compuesto por múltiples nodos independientes que proporcionan datos. Esta descentralización elimina el punto único de falla presente en un oráculo centralizado.

Al tener múltiples fuentes de datos independientes, un oráculo distribuido proporciona mayor confiabilidad y transparencia en los datos obtenidos, ya que, en un oráculo centralizado, existe el riesgo de que de manipulación o falsificación de la información. Un oráculo distribuido reduce significativamente este riesgo, ya que se requeriría que estuvieran comprometidos numerosos nodos independientes (Beniiche, 2020).

Frecuentemente los oráculos distribuidos utilizan mecanismos de consenso y criptografía para garantizar la integridad de los datos proporcionados, y dificultar la manipulación o alteración de los datos. Con la finalidad de fomentar la honestidad y la precisión de los datos, algunos oráculos distribuidos implementan sistemas de incentivos y reputación para los nodos que proporcionan datos no erróneos o engañosos, en caso contrario son penalizados. Los oráculos distribuidos suelen ser más interoperables e integrarse con diferentes plataformas y redes Blockchain, lo que facilita su adopción en diversos escenarios y aplicaciones. Los oráculos descentralizados evitan la concentración de información sensible en un solo punto, y dependiendo de la implementación, pueden proporcionar mayor privacidad y protección de datos en comparación con los oráculos centralizados.

Aunque los oráculos distribuidos pueden ser más complejos y costosos de implementar en comparación con los oráculos centralizados, las ventajas en términos de confiabilidad, transparencia y resistencia a fallas los convierten en una opción más adecuada para escenarios críticos como la gestión de SLA en entornos basados en BC.

3.6 API expuesta por los proveedores

En el contexto de las redes 5G, un proveedor de NS puede exponer una API para el monitoreo de NS a través de la NEF del núcleo de 5G. Esta NF actúa como un punto de entrada seguro para que las aplicaciones externas puedan tener acceso a las capacidades y servicios de la red 5G, incluyendo el monitoreo de NS (3GPP, 2019). El proveedor de NS define la API que desea exponer, la cual puede incluir operaciones para el monitoreo de métricas de rendimiento, estadísticas de uso, eventos y alertas relacionados con los NS. También registra su API de monitoreo en la NEF, proporcionando información como la descripción de la API, los puntos finales (*endpoints*), los métodos permitidos y cualquier otro detalle relevante.

La NEF implementa mecanismos de autenticación y autorización para controlar el acceso a las API

expuestas. Los sistemas de monitoreo externos o las aplicaciones de terceros deben autenticarse y obtener los permisos necesarios para poder acceder a las API de monitoreo. Cuando un sistema externo realiza una solicitud a la API de monitoreo, la NEF actúa como un proxy, enrutando la solicitud al componente correspondiente del núcleo de 5G que gestiona el monitoreo de los *Network Slices*.

3.7 Arquitectura propuesta para la automatización del ciclo de vida del SLA.

Para el establecimiento, automatización y ejecución de los SLA de NS, y como objetivo de este trabajo de investigación, se propone la arquitectura que en esta sección se trata a detalle en cuanto a las características y funcionamiento de cada componente que la conforman, así como las interacciones entre estos.

La operación del esquema propuesto se puede segmentar en 7 procedimientos o fases principales y automatizar así el ciclo de vida del SLA:

- Registro de proveedores: Se hace un proceso donde se registran los proveedores para poder ofertar sus NS.

- Descubrimiento de proveedores: Mediante una solicitud a la dApp se les muestra a los clientes quiénes son los proveedores de NS.

- Definición de SLA: Los proveedores hacen una propuesta de SLA donde especifican los umbrales de los parámetros de desempeño del NS.

- Descubrimiento de SLA: A partir de una solicitud a la dApp se les hace saber a los clientes los SLA disponibles.

- Establecimiento de SLA: Luego de un proceso de validación queda establecido el acuerdo entre cliente y proveedor.

- Monitorización: Por medio de un oráculo distribuido se obtienen las métricas de desempeño del NS y se determinan las violaciones o incidencias.

Terminación y cumplimiento de penalidades: Se finaliza el contrato al llegar su fecha de expiración y a partir de las incidencias se hacen los pagos y compensaciones a proveedor y cliente respectivamente.

A continuación, se explican cada uno de estos procedimientos. Estos representan una versión más específica de la lógica del esquema, que en su totalidad puede ser llevada a cabo mediate la ejecución de códigos y escrita en diferentes lenguajes de programación.

3.7.1 Registro de proveedores

Se observa el diagrama de secuencia del registro de proveedores (Figura 16) para la arquitectura de gestión de SLA propuesta, el registro de proveedores comienza con una solicitud a la dApp de un proveedor para añadir un nuevo proveedor. Esta solicitud requiere como parámetros la dirección de monedero digital del nuevo proveedor y el identificador o nombre del proveedor. La dApp reenvía esta solicitud a SC Mercado el cual limita la ejecución de la función para agregar proveedores; solo un proveedor que se encuentre registrado puede añadir un nuevo proveedor.

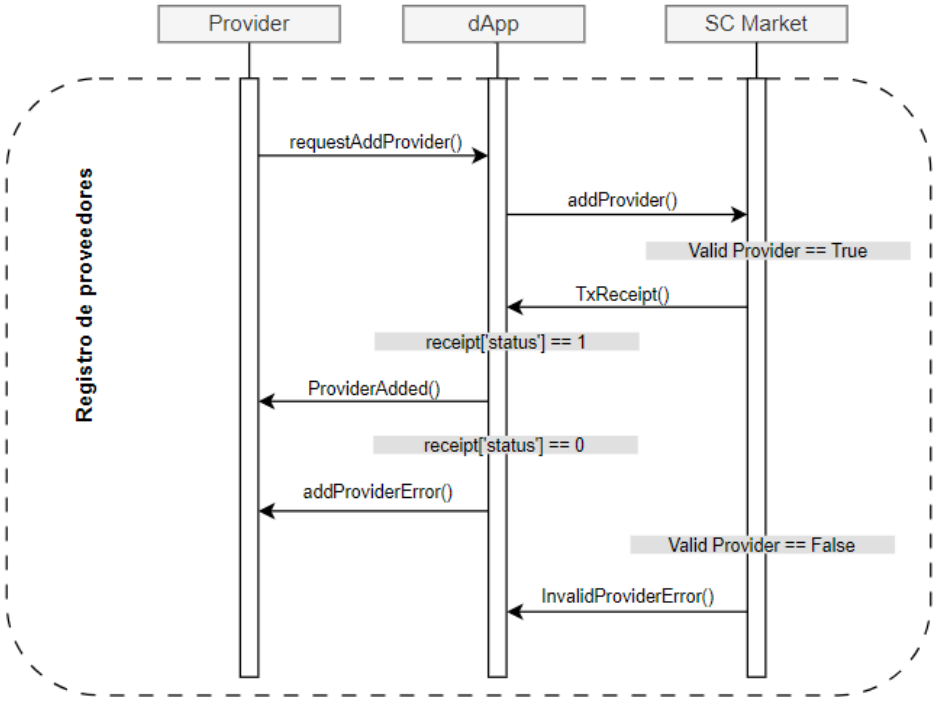

Figura 16. Diagrama de secuencia de registro de proveedores [Elaboración Propia].

Cuando SC Mercado reciba esta solicitud, comprobará si es válido el proveedor que está haciendo la solicitud. De ser correcto, SC Mercado envía un recibo de transacción (TxReceipt) a la dApp que contiene

la información del estado de la solicitud (receipt[status]). Si el estado es “1” se agregó correctamente el proveedor, y es “0” en caso contrario. En ambos casos se notifica al proveedor del resultado de su solicitud desde la dApp. La adición de este nuevo proveedor genera el lanzamiento de un evento que se registra en la BC llamado ProviderAdded. Este registro se podrá acceder vía la dApp en cualquier momento sin costo. Si un usuario que no es proveedor intenta registrar un proveedor nuevo SC devolverá el error InvalidProviderError().

Los eventos en una BC son acontecimientos específicos que ocurren en la red y que pueden ser capturados y registrados para su procesamiento posterior. Estos eventos pueden ser generados por contratos inteligentes, transacciones, cambios en el estado de la BC u otras acciones relevantes que ocurren en el sistema. Por otro lado, los logs de eventos son registros o entradas que se crean cuando se produce un evento específico. Estos logs se almacenan en la cadena de bloques y pueden ser consultados y accedidos por los usuarios y las aplicaciones para obtener información sobre eventos pasados. Es importante destacar que los eventos se utilizan comúnmente para notificar a los usuarios y a las aplicaciones sobre cambios o actividades específicas en la BC de forma asíncrona, lo que significa que no necesariamente las aplicaciones estén constantemente consultando la BC para detectar cambios.

De forma similar al registro de proveedores se realiza el registro de clientes, pero en ese caso se registra el propio cliente y solo se puede registrar él mismo.

3.7.2 Descubrimiento de proveedores

El proceso para el descubrimiento de proveedores por parte del cliente se aprecia en el diagrama de secuencia de la Figura 17. Un cliente puede solicitar el descubrimiento de proveedores a la dApp, para ello el cliente debe estar previamente registrado de no ser así recibirá la notificación de error *requestDscoveryError()*.

La solicitud se maneja por la dApp la cual va a estar suscrita al evento *ProviderAdded* del SC Mercado. Al estar suscrita al evento permite hacer lecturas de los registros, por tanto, maneja la solicitud del descubrimiento de proveedores haciendo una lectura del registro de proveedores añadidos (*readProviderAddedEventLogs*). SC Mercado devuelve los datos solicitados (*retrieveProviderAddedLogs*) a la dApp y ésta los muestra al cliente que realizó la solicitud.

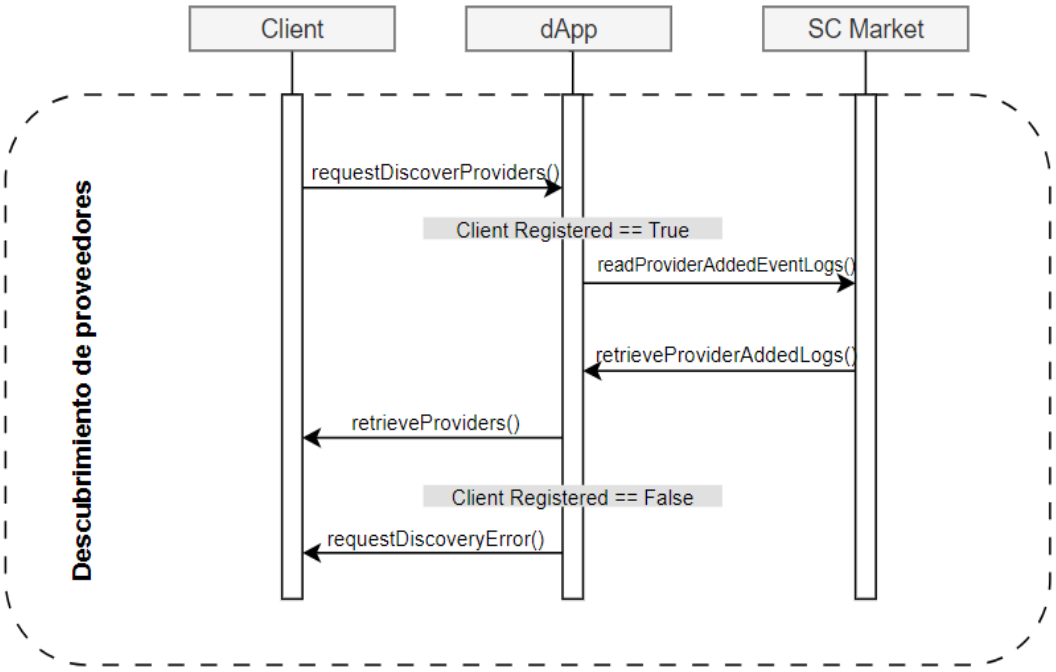

Figura 17. Diagrama de secuencia de descubrimiento de proveedores [Elaboración Propia].

3.7.3 Definición de SLA

El proceso de definición de SLA comienza por una propuesta de SLA que hace un proveedor a la dApp como se muestra en la Figura 18. Esta propuesta va acompañada del archivo del documento legal que refleja todas las condiciones del SLA, los umbrales de parámetros de monitoreo para el NS en cuestión, el periodo de medición a utilizar, el tiempo de duración del contrato, los *endpoints* de la API del proveedor y el intervalo de duración de la subasta asociada al SLA propuesto. La dApp validará la propuesta con respecto al formato adecuado de la misma.

Si la propuesta es válida, la dApp se encarga de subir a la red IPFS el documento SLA y la red IPFS envía un *hash* como confirmación de la recepción correcta del documento para su almacenamiento. Este *hash* podrá ser utilizado posteriormente para recuperar el documento de ser necesario. Con el *hash* y el resto de la información de la propuesta la dApp solicita la creación de un SLA al SC Mercado el cual retorna un recibo de transacción con el estado de la transacción. Con un estado de la transacción recibido como correcto, va la información relativa a la adición del SLA la cual es conservada en la dApp agregando el SLA al perfil del proveedor en la dApp. Con la creación de un SLA se emite el evento *CreateSLA.* En caso de que la propuesta no sea válida, se rechaza la solicitud.

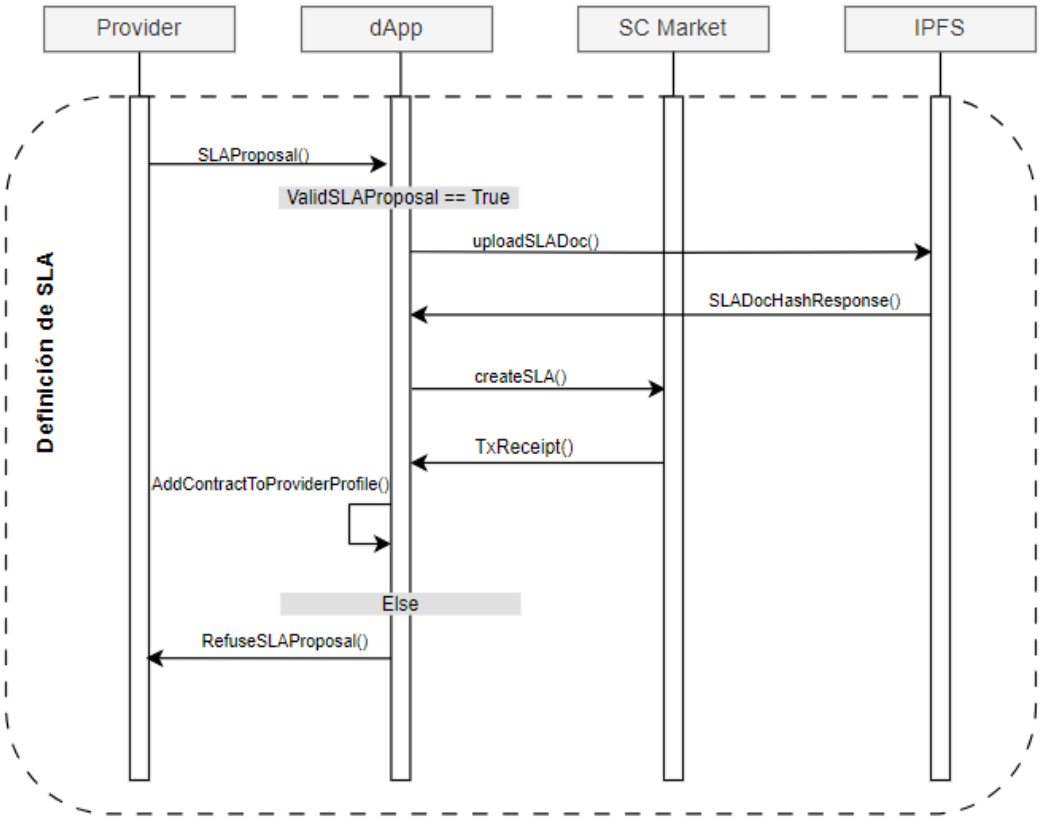

Figura 18. Diagrama de secuencia de definición de SLA [Elaboración Propia].

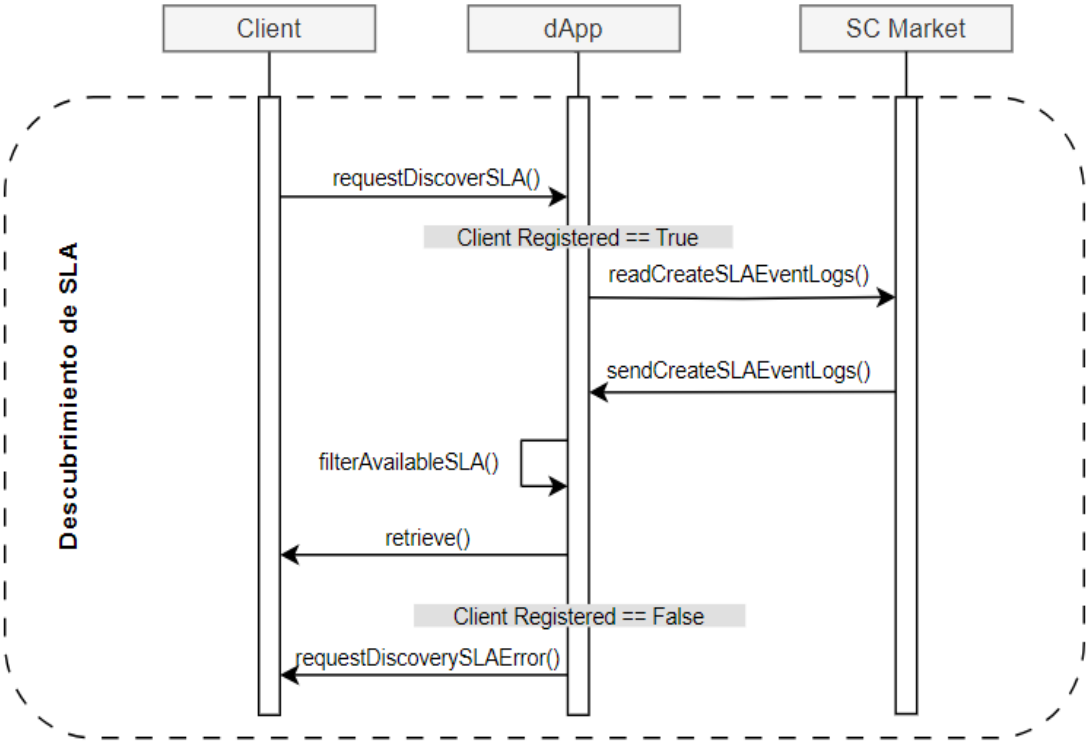

Figura 19. Diagrama de secuencia de descubrimiento de SLA [Elaboración Propia].

3.7.4 Descubrimiento de SLA

El procedimiento para el descubrimiento de SLA aparece como diagrama de secuencia en la Figura 19. En este diagrama se puede apreciar que tiene un comportamiento similar al descubrimiento de proveedores analizado anteriormente. En este caso los registros de evento consultados son los correspondientes al evento CreateSLA.

Una vez recibidos (retrieve) los logs de este evento, la dApp los filtra para mostrar al cliente solamente los SLA disponibles, aquellos cuya fecha de expiración no haya sido excedida, para que el cliente pueda hacer ofertas sobre un SLA en específico. Luego de este filtrado de los SLA disponibles, se comparte con el cliente la dirección pública del SC Subasta correspondiente para que puedan hacer sus ofertas por el SLA que les parezca de interés.

3.7.5 Establecimiento de SLA

El proceso de establecimiento de SLA es más complejo que los anteriores. Este se puede observar en la Figura 20. Ocurre como continuación de la fase de definición del SLA iniciada por el proveedor, pero es necesario que este SLA creado tenga asociado un cliente. Para ello como respuesta a la solicitud de creación de un nuevo SLA, el SC Mercado despliega SC SLA que estará inactivo y un SC Subasta vinculado directamente al SLA que gestionará el proceso de subasta del SLA.

Un contrato inteligente tiene la capacidad de desplegar otro contrato inteligente, siempre que tenga el código fuente. Esto lo hace llamando a la función constructor() en el código fuente del SC que se desee desplegar. En el caso de SC SLA se llama a la función constructor() asignándole los parámetros de: nombre del proveedor, la dirección de monedero digital del proveedor, el hash del documento SLA correspondiente en la red IPFS, los umbrales de los parámetros de monitoreo, así como, los endpoints de la API del proveedor. Al realizar el despliegue de un contrato se reciben entre otros parámetros, la dirección pública que se le asigna a este en la BC. La dirección pública de SC SLA la utiliza el SC Mercado y la pasa como parámetro cuando se llama a la función contructor() de SC Subasta.

En la llamada de la función constructor de SC Subasta, además de la dirección de SC SLA, se pasa como parámetros: el intervalo de subasta, la dirección del proveedor (que define el beneficiario) y el valor mínimo de oferta. A partir de ese momento puede recibir ofertas de los clientes, los cuales pueden saber

la dirección pública del SC Subasta vinculado al SLA por medio de la dApp mediante el descubrimiento de proveedores explicado anteriormente.

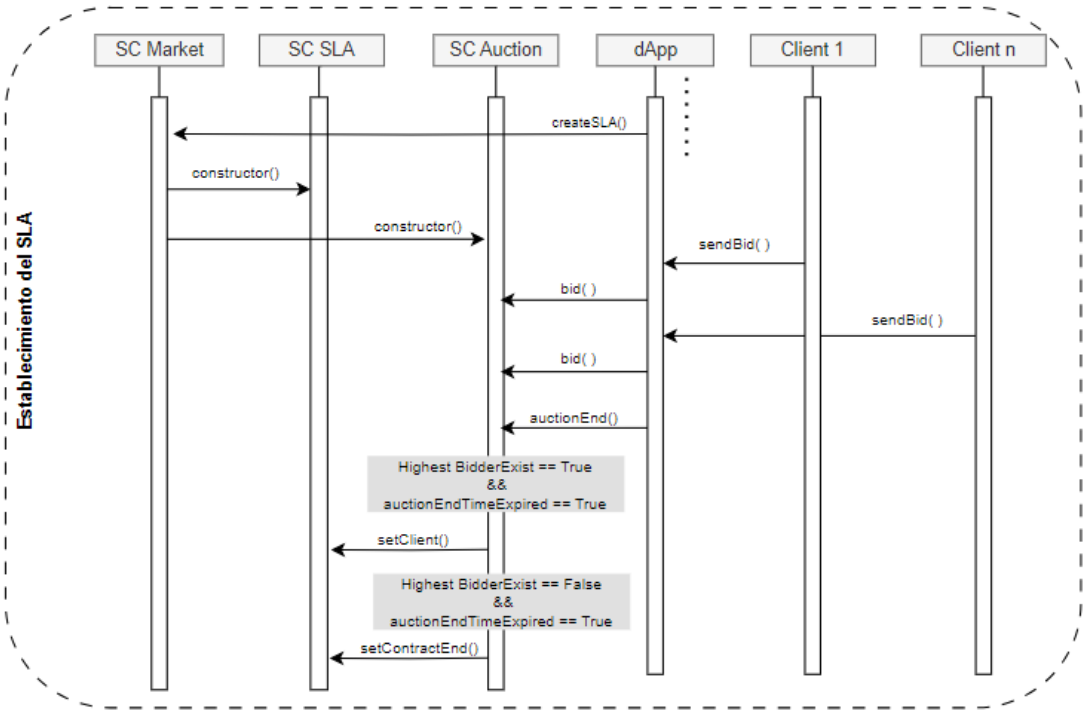

Figura 20. Diagrama de secuencia de establecimiento del SLA [Elaboración Propia].

La dApp utilizará un temporizador para definir el fin de la subasta llamando a la función AuctionEnd() del SC Subasta. La llamada de esta función también se pudiera realizar desde un oráculo externo, sin embargo, en el esquema propuesto ocurre desde la dApp. En la llamada a la función se le pasa como parámetro el valor de la oferta que será transferido al contrato. Luego de culminar la subasta se determina si hubo una oferta ganadora, de ser así, se activa el SC SLA correspondiente asignándole un cliente que será el mayor postor, y finalmente se le transfiere su pago al SC SLA.

3.7.6 Monitorización

Como último paso en el proceso de establecimiento del SLA, se inicializa un elemento que forma parte del oráculo distribuido, el cual es capaz de hacer llamadas periódicas a una función de un contrato. En la Figura 21 el oráculo distribuido utilizado es *Chainlink*, y el elemento que se crea para posibilitar llamadas periódicas es *Chainlink Keepers* (Chainlink, 2024a), mediante la función *SetChainlinkKeeper()*. Se seleccionó *Chainlink* como por ser compatible con la BC a utilizar en la implementación.

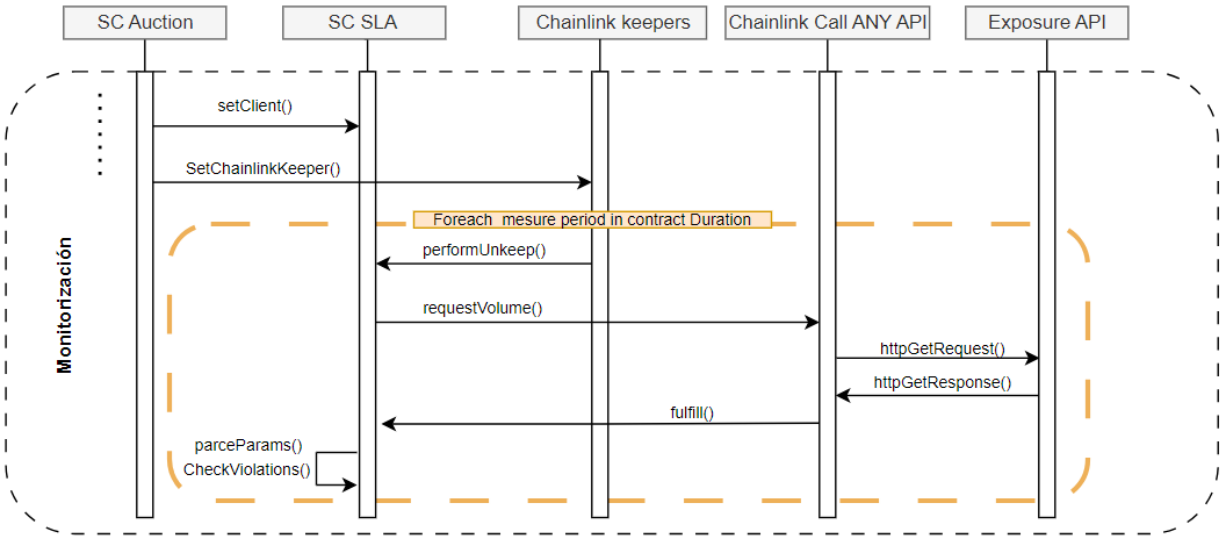

Figura 21. Diagrama de secuencia de monitorización del SLA [Elaboración Propia].

Para iniciar se le debe enviar al *Chainlink keeper*, tanto la dirección de SC SLA, como el periodo que se utiliza para llamar continuamente a la función objetivo. Estos parámetros los necesita para hacer las llamadas periódicas. La lógica para enviar los datos e inicializar el *Chainlink Keeper* se implementa en SC Subasta.

Debido a las limitaciones inherentes de las BC que dan soporte a SC, se requiere que una función de un contrato inteligente sea llamada para ejecutarse periódicamente por un agente externo. Los SC no pueden ejecutar tareas de forma automática o programada si no son llamados explícitamente por una transacción externa. No tienen una noción interna del tiempo o la capacidad de programar eventos recurrentes por sí mismos.

Esta limitación existe por diseño, ya que los SC deben ser deterministas y ejecutarse de manera idéntica en todos los nodos de la red BC. Si los contratos pudieran programar o disparar tareas de forma autónoma, sería difícil mantener la sincronización y el consenso en la red. Por esta razón, si se necesita que una función de un contrato inteligente se ejecute periódicamente se depende de un agente externo que llame a esa función en los intervalos deseados.

La función que será llamada de forma periódica por *Chainlink Keeper* es *performUnkeep()* la cual en su lógica interna ejecuta la función *requestVolume()* de SC SLA. Esta función construye una solicitud para *Chainlink ANY API*, que es otro servicio de *Chainlink*, donde utiliza los *endpoints* configurados en la propuesta de SLA para que *Chainlink ANY API* extraiga la información de las métricas de monitorización de

la API del proveedor usando una solicitud http (httpGetRequest()).

Una vez que *Chainlink ANY API* tenga la información solicitada, *httpGetResponse()* la reenvía a SC SLA llamando a la *función fullfill()*, a la que pasa la información como parámetros de la función. Al recibir los datos de las métricas de monitoreo de NS, con el fin de determinar si hubo violaciones o no, el SC SLA formatea los datos (*parseParams()*), lo que permite según corresponda a cada métrica, hacer las operaciones pertinentes y comprobar con la función *CheckViolations()* si se excedieron o no los parámetros de QoS contratados.

3.7.7 Terminación y cumplimiento de penalidades

En la monitorización, la llamada de *Chainlink Keeper* se realiza para cada periodo de medición en el tiempo de duración del contrato; cuyas funciones terminan al culminar el contrato y se debe eliminar, de lo cual se encarga SC SLA al ejecutar la función *deleteChainlinkKeeper()*. El proceso de terminación y cumplimiento de penalidades se muestra en la Figura 22. Auto seguido el SC SLA calcula las penalidades que se aplicarán al proveedor por la detección de incidencias en las métricas obtenidas.

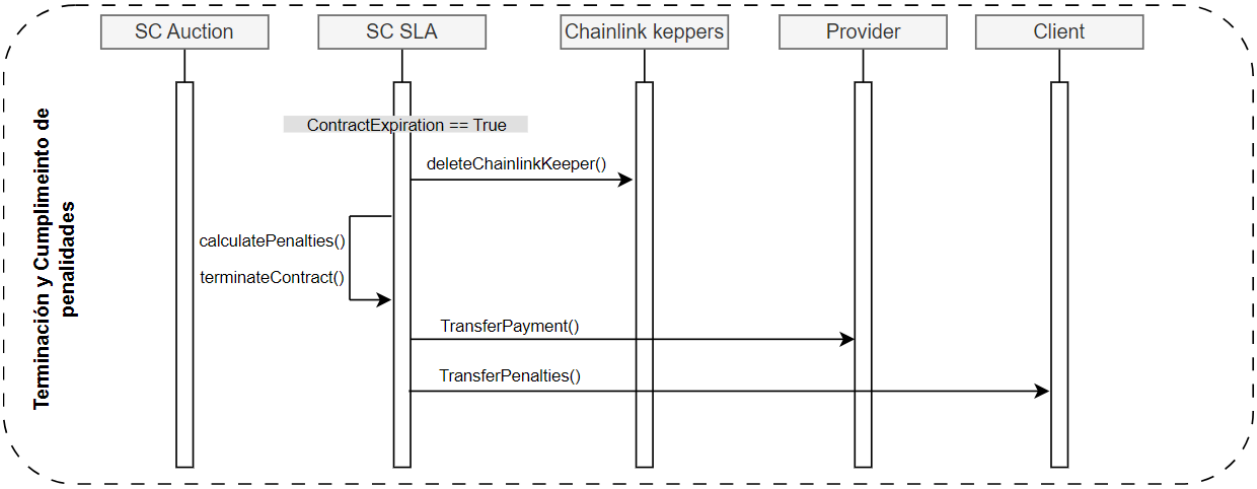

Figura 22. Diagrama de secuencia de terminación y cumplimiento de penalidades [Elaboración Propia].

El cálculo de las penalidades puede estar basado en el porcentaje de disponibilidad del servicio definido por el porciento de la cantidad de mediciones que no representan una violación sobre total de mediciones durante la vigencia del contrato. El cálculo y la implementación de las compensaciones son específicas según las cláusulas de los SLA. En este trabajo se proporciona una prueba de concepto (PoC, por las siglas

del término en inglés, *Proof of Concept*) donde se trata la disponibilidad como el indicador principal. En la PoC implementada para validar el esquema propuesto, la compensación es del 10% del pago cuando se logra una disponibilidad entre 90% y 99% y para una disponibilidad menor al 90% corresponde un 30% del pago.

3.8 Recomendaciones de implementación

Para la implementación de la propuesta planteada, se deben analizar qué tecnologías utilizar, en base a criterios de selección que doten al esquema de las mejores características posibles. La elección de las tecnologías es fundamental para garantizar el éxito, y el rendimiento óptimo del esquema, y se debe de tener en cuenta la capacidad de integración y la interoperabilidad entre los diferentes componentes del mismo. Con el esquema planteado se persigue proporcionar una arquitectura segura, confiable, transparente, distribuida y sin la dependencia de terceros que asegure el desempeño necesario para llevar a cabo la gestión automatizada de SLA de NS. Estas características deben ser la base para definir los componentes del esquema.

Para completar el esquema se debe seleccionar una forma de despliegue para el *frontend* de la dApp y una BC con capacidades para ejecutar SC que funcione como *backend* de la dApp. Otro elemento esencial, es garantizar la conexión a la BC seleccionada, ya sea implementando nodos propios o utilizando proveedores de nodos. Con el fin de poder entregar datos y posibilitar la interacción con sistemas externos a la BC, se debe utilizar algún tipo de oráculo. En las secciones siguientes se exponen los criterios y consideraciones para la selección de una u otra tecnología para cada uno de los casos anteriores, y satisfacer los requisitos del esquema de gestión de SLA planteado.

3.8.1 Consideraciones para implementación de la dApp

Existen varias formas de desplegar y almacenar el *frontend* de una dApp. Una opción común es usar un servidor Web centralizado, que es la forma tradicional de alojar aplicaciones Web. Para este caso, el código fuente del *frontend* se almacena en un servidor Web centralizado y se sirve a los usuarios a través de HTTP o HTTPS. Esta opción es fácil de configurar y mantener, pero va en contra del principio de descentralización de las dApps.

Otra manera de desplegar y almacenar el *frontend* de una dApp es mediante un sistema de archivos descentralizados. Una opción excelente es IPFS, ya que distribuye y replica los archivos en múltiples nodos de una red descentralizada, lo que se garantiza la disponibilidad y no hay puntos únicos de falla. Aunque hay varias opciones para desempeñar el papel de sistema de archivos descentralizado como Storj o Swarm (Huang et al., 2020), estos no presentan la madurez de IPFS, que es un sistema con escalabilidad y buen rendimiento, así como una comunidad activa y diversa que contribuye a su desarrollo y adopción.

Para el *backend* se debe utilizar una BC, que por medio de los SC lleve a cabo los procedimientos que se especifican en el esquema propuesto, donde los SC se encargan de toda la operación para garantizar una gestión eficiente de los SLA de NS. No todas las BC tienen soporte para SC por lo tanto se debe seleccionar una que si tenga esta capacidad. Por ejemplo, el caso de Bitcoin representa una BC que no admite la programación de SC.

Para el *backend* se habrá de elegir entre una BC pública o privada lo cual dependerá de factores, como el nivel de descentralización, transparencia, escalabilidad, rendimiento, privacidad, control de acceso y costos requeridos. Ambos enfoques presentan ventajas y desafíos que deben evaluarse cuidadosamente.

Las BC públicas ofrecen un alto grado de transparencia, ya que la información es accesible por todos los nodos que conforman la red, incluyendo clientes y proveedores. Al carecer de una autoridad centralizada, son más resistentes a la manipulación o intenciones maliciosas que puedan favorecer a una parte específica(S. Zhou et al., 2023) . Cualquier interesado puede realizar verificaciones y auditorías debido a la naturaleza pública de los datos. Sin embargo, estas BC conllevan costos operativos altos, ya que se debe pagar tarifas por cada transacción realizada. Además, pueden implicar costos de infraestructura, ya que se necesitan nodos propios para interactuar con la red pública o pagar a proveedores de nodos por sus servicios (P. Zheng et al., 2023). Su rendimiento y escalabilidad son más limitados debido a los mecanismos de consenso utilizados y al número de nodos participantes, lo que puede consumir más recursos y tiempo para ejecutar transacciones, especialmente debido a que, cuando crece la red es más complejo, se dificulta alcanzar al consenso en la conformación de un nuevo bloque.

Por otro lado, las BC privadas suelen ofrecer un mejor rendimiento y mayor escalabilidad, ya que involucran un número reducido de nodos y utilizan mecanismos de consenso más eficientes, sin la necesidad de realizar cálculos complejos para competir por recompensas, como ocurre en algunas BC públicas. Al mismo tiempo, las BC privadas brindan mayor privacidad al restringir el acceso a los datos a fuentes distintas de clientes y proveedores, lo que es una ventaja en el esquema propuesto ya que los

proveedores pueden preferir mantener la información de su gestión privada. En una BC privada, los costos por operaciones no son requeridos, lo que representa un ahorro considerable. Sin embargo, existen costos asociados al mantenimiento de la infraestructura en términos de hardware y software necesarios para conformar la red privada.

Se puede resumir que la elección entre una BC pública o privada dependerá de las prioridades específicas del esquema de gestión propuesto que se puedan asumir, como el nivel de transparencia deseado, los requisitos de rendimiento y escalabilidad, la privacidad requerida, y los costos operativos y de infraestructura. En la Tabla 3 se muestra una comparación entre BC pública y privada con respecto a su utilización como backend de la dApp para el esquema propuesto.

Tabla 3. Comparación entre BC pública y BC privada en su uso como *backend* [Elaboración propia].

Característica	BC Pública	BC Privada
Escalabilidad y rendimiento	Baja	alta
Descentralización y transparencia	Alta	baja
Costos Operativos	Altos	bajos
Costos de Infraestructura	Medios	altos
Privacidad	Baja	alta

El caso de la elección de una BC pública en específico, está determinada por los requisitos de escalabilidad, costo y tiempo de las transacciones, su grado de descentralización, su grado de madurez y adopción, el tamaño de su comunidad y el nivel de soporte que exista para la misma. Presentan diferentes mecanismos de consenso y requieren determinados lenguajes de programación para los SC. En la Tabla 4 se comparan varias BC según el tipo de mecanismo de consenso y lenguaje que utilizan para el desarrollo de los SC.

Para el caso de implementación de una BC privada o permisionada para el *backend* de la dApp existen varias tecnologías que se pueden utilizar como, *Hiperledger Fabric, Corda, Quorum* y *Binance Smart Chain*. En la Tabla 5 se puede ver una comparación entre ellas en cuanto a mecanismo de consenso y lenguaje de programación de SC utilizado.

Tabla 4. Comparación entre plataformas Blockchain Públicas [Elaboración propia].

Red Blockchain	Mecanismo de	Lenguaje de programación de
Ethereum	PoS	Solidity
Polygon	PoS	Solidity
VeChain	PoA	Solidity
EOS	dPoS	C++
Tron	dPoS	Solidity
Solana	PoH+PoS	Rust
Cardano	Ouroboros	Plutus

Tabla 5. Comparación entre plataformas BC permisionadas [Elaboración propia].

Red Blockchain	Mecanismo de consenso	Lenguaje de programación de
Hyperledger Fabric	Raft, IBFT y otros	Go, Node.js, Java, Solidity y
Corda	Corda consensus	Kotlin, Java
Quorum	Raft e IBFT	Solidity
Binance Smart Chain	PoA+PoS	Solidity

También existen plataformas de proveedores de nube que se utilizan con el fin de implementar una BC privada o permisionada para el backend de la dApp como, por ejemplo, Microsoft Azure Blockchain Service, Amazon Quantum Ledger Database, Google Cloud Blockchain Platform e IBM Blockchain Platform. El tipo de consenso y lenguajes que se pueda utilizar es determinado por las configuraciones de estas herramientas.

Las plataformas de proveedores de nube, como *Microsoft Azure, Amazon Web Services* (AWS), *Google Cloud* e *IBM Cloud*, ofrecen servicios de BC privada o permisionada, simplifican el despliegue, la gestión y el mantenimiento de la infraestructura necesaria. Estas plataformas en la nube ofrecen características adicionales, como herramientas de desarrollo, gestión de identidades, monitoreo, escalabilidad automática y alta disponibilidad. Además, permiten la integración con otros servicios en la nube, como

bases de datos, servicios de mensajería y herramientas de análisis, lo que facilita la creación de soluciones completas y complejas basadas en BC privadas o permisionadas.

3.8.2 Opciones de conexión a la Blockchain

Para que una dApp se conecte a una BC y sea capaz de llevar a cabo transacciones, es necesario interactuar con uno o varios nodos de la red. De forma general una dApp utiliza el protocolo de Llamada de Procedimiento Remoto (RPC), para establecer una comunicación remota con estos nodos y ejecutar procedimientos en ellos.

El protocolo RPC permite que la dApp invoque métodos o funciones en un nodo remoto de la BC, como si estuviera llamando a un procedimiento local. Esto se logra a través de una URL RPC que apunta al nodo con el que la dApp desea interactuar. RPC funciona bajo un modelo cliente-servidor, donde el software que envía la solicitud es el cliente y el software basado en una máquina diferente es el servidor que responde a la solicitud (Weihl, 1990).

Se puede configurar la dApp para que ejecute su propio nodo completo en la BC, lo que le permitiría interactuar directamente con la red sin depender de terceros. Sin embargo, esto requiere recursos computacionales significativos y una configuración compleja. La implementación de un nodo ligero, que solo almacena una parte de la cadena de bloques y confíe en nodos completos para obtener datos adicionales, puede ser una opción más ligera para una dApp con requisitos menos intensivos.

Otra solución sería utilizar los servicios de proveedores de nodos que ofrecen acceso a nodos de BC a través de URL RPC. Estas plataformas proporcionan nodos confiables y escalables, lo que simplifica la conexión para las dApps por medio de una API para llevar a cabo las transacciones (Leal, 2024).

Correr un nodo propio proporciona mayor control, privacidad y seguridad sobre las operaciones que se realicen ya que sería administrado por los gestores de la dApp, pero tiene limitaciones en cuanto a escalabilidad y costos y tiempo invertido en el mantenimiento de infraestructura en comparación al uso de un proveedor de nodos. La elección de uno u otro enfoque debe estar basada en los requisitos específicos de la dApp. En la Tabla 6 se muestra una tabla comparativa entre el uso de nodos propios completos o ligeros y el uso de servicios de proveedores de nodos.

Los proveedores de nodos ofrecen un conjunto de servicios y funcionalidades clave para facilitar la integración de aplicaciones descentralizadas dApp con BC. Además de proporcionar acceso a nodos de BC a través de URL RPC, ofrecen infraestructura escalable y de alta disponibilidad para manejar un gran volumen de solicitudes, así como bibliotecas de lenguajes de programación y kit de herramientas para el desarrollo de software (SDK, por las siglas del inglés, *Software Development Kit*) que simplifican la integración.

Tabla 6. Comparativa entre uso de nodos propios y uso de proveedores de nodos [Elaboración Propia].

Aspecto	Nodos propios	Proveedores de nodo
Control	Mayor control y soberanía sobre los nodos.	Se depende de un tercero para proporcionar el acceso a los nodos.
Flexibilidad	Se pueden personalizar los nodos según los requisitos de la dApp	Limitado a configuraciones específicas del proveedor de nodo
Privacidad	Tiene mayor privacidad pues no revela su actividad de terceros	Toda la actividad es visible para los proveedores de nodos
Costos	Costos asociados con el hardware, la infraestructura y el mantenimiento de los nodos.	Los proveedores de nodos tienen una capa gratuita para sus servicios, pero cobran para casos de uso de producción a gran escala.
Escalabilidad	El administrador de la dApp administra la escalabilidad de los nodos según la demanda	Los proveedores pueden escalar fácilmente sus recursos para manejar una mayor carga

Los proveedores de nodos también brindan servicios de monitoreo, análisis, almacenamiento y gestión de datos relacionados con la BC. Algunos proveedores se integran con otros servicios en la nube y ofrecen soporte técnico, documentación y medidas de seguridad robustas. Si bien estos servicios son convenientes, también introducen una dependencia de terceros y posibles implicaciones en términos de privacidad y descentralización. En la Tabla 7 se observa una comparación entre los diferentes planes de la capa gratuita de los proveedores de nodos *Alchemy, Infura y Quicknode*.

Tabla 7. Comparación de la capa gratuita de servicio de proveedores de nodo [Elaboración propia].

Características	*Alchemy*	*Infura*	*Quicknode*
Límite de solicitudes por segundo	9 rps	10 rps	16.5 rps
Límite solicitudes por periodo	8 108 108 mes	100 000 día	2 500 000 mes
API Keys disponibles	5	1	1

Como se puede apreciar cada proveedor de nodo tendrá cierta capacidad para la cantidad de solicitudes por segundo que puede manejar en un periodo de tiempo. Por lo general para a los clientes de proveedores de nodo se le otorgan credenciales de acceso conocidas como API *keys* que le permiten autenticarse y hacer llamadas URL RPC por medio de las API del proveedor de nodos. Cada API *key* se utiliza para una BC en específico. En la Tabla 7 se aprecia que *Alchemy* es la que más solicitudes acepta mensualmente y tiene un mayor número de API *key* mientras que *Quicknode* soporta más solicitudes por segundo. En dependencia de los requisitos será la elección de uno u otro proveedor, si se desea dar cobertura a gran cantidad de solicitudes durante todo un mes *Alchemy* es el mejor candidato (para el esquema propuesto), pero si la prioridad es la cantidad soportada diariamente *Quicknode* es más eficaz.

Sea con un nodo propio o por medio de un proveedor de nodos al tener acceso a una URL RPC, para enviar transacciones al nodo, la dApp puede utilizar bibliotecas de cliente específicas de la BC en cuestión (por ejemplo, Web3.js para *Ethereum* o *Fabric SDK* para *Hyperledger Fabric*). Estas bibliotecas abstraen la complejidad de interactuar directamente con la BC y proporcionan una interfaz de programación más accesible. Las transacciones enviadas por la dApp a través del nodo serán propagadas a toda la red BC, donde serán validadas y finalmente incluidas en un nuevo bloque según los mecanismos de consenso de la BC específica.

Es importante tener en cuenta que, para garantizar la integridad y la descentralización, se recomienda que las dApps se conecten a múltiples nodos de la BC, en lugar de depender de un solo punto de falla. Además, las transacciones requieren el pago de tarifas de *gas* o comisiones en el caso de las BC públicas, y deben ser firmadas con una clave privada válida para su aceptación y procesamiento por la red BC. Para las BC privadas, que son controladas y mantenidas por una entidad específica, los participantes suelen tener roles definidos y acceso controlado a los recursos de la BC. En lugar de una billetera digital que gestione criptomonedas, los participantes pueden interactuar con la BC utilizando interfaces o API proporcionadas por la plataforma de BC privada. Estas interfaces permiten a los usuarios enviar transacciones, tener acceso a datos específicos o realizar otras operaciones autorizadas según sus roles y permisos en la red.

3.8.3 Selección de oráculo distribuido

El hecho de que la BC sea pública o privada es un factor relevante para la selección de un oráculo para la interacción con fuentes externas en el contexto de la gestión de SLA y SC. En el caso de BC públicas, como Ethereum, donde cualquier persona puede unirse y participar, es fundamental utilizar oráculos descentralizados y confiables. Estos oráculos, como *ChainLink*, *Band Protocol*, *API3* o *Provable*, están diseñados para proporcionar datos de fuentes externas de manera segura y verificable, sin depender de una sola entidad centralizada lo cual es crucial en un entorno abierto y descentralizado.

A continuación, se presentan una breve descripción de algunos oráculos que se pueden utilizar en el caso de BC públicas:

- *Chainlink*: Es la primera solución de oráculos descentralizada altamente confiable en la red Ethereum. Está compuesto por varios nodos oráculos y fuentes de datos descentralizadas, múltiples métodos de agregación de datos, computación privada y fuera de línea mediante un TEE, un sistema de reputación, un mecanismo de recompensas y penalizaciones, y firmas de umbral. Sin embargo, presenta algunas limitaciones, como los altos costos de agregación en cadena, poca escalabilidad y la posibilidad de centralización del sistema basado en reputaciones (Lin et al., 2022).

- *Provable* (antes Oraclize): Es un oráculo centralizado que proporciona pruebas criptográficas de autenticidad para los datos obtenidos de fuentes externas. Ofrece una solución sencilla y rentable para algunas aplicaciones, aunque al ser una solución centralizada trae consigo el problema de un solo punto de falla (Beniiche, 2020).

- *Band Protocol* es una solución de Oráculo descentralizado que emplea un mecanismo de apuesta y votación, donde los nodos validadores son seleccionados de manera aleatoria y deben comprometer tokens para participar. Utiliza una arquitectura de capa 2 que separa la recopilación de datos de la ejecución en cadena, lo que le otorga un mayor rendimiento y escalabilidad (Lin et al., 2022). A pesar de sus ventajas tiene poca adopción por la comunidad y poca documentación para integrarlo en las aplicaciones con respecto a Chainlink.

Se propone utilizar *Chainlink* como oráculo para la implantación del esquema propuesto ya que es una solución con mayor adopción y madurez que cuenta con una gran comunidad de desarrolladores, una

extensa documentación y un sólido soporte técnico. Está integrado en numerosos proyectos y BC importantes, como *Ethereum*, *Binance Smart Chain*, *Polygon,* entre otros y está demostrado que es una solución segura y confiable a lo largo del tiempo, superando auditorías de seguridad rigurosas. Su arquitectura descentralizada y el sistema de reputación de nodos brindan resistencia a la manipulación y puntos únicos de falla. *Chainlink* también presenta una amplia gama de integraciones y funcionalidades como oráculos que hacen llamadas externas u otros que posibilitan la automatización en la ejecución de funciones de SC (Chainlink, 2024a) . *Chainlink* es compatible con *Ethereum* y *Polygon*. Dado que en la implementación del esquema se utilizarán estas BC es conveniente utilizarlo como oráculo distribuido.

En el contexto de BC privadas o permisionadas, donde el acceso está restringido a participantes conocidos, existe mayor flexibilidad en cuanto a la elección del oráculo. En estos casos, se pueden utilizar oráculos más simples o incluso soluciones personalizadas, ya que el acceso a la información y la confianza en los proveedores de datos están controlados de manera más estricta. En BC privadas es más factible la integración de oráculos que utilicen API o fuentes de datos propiedad de la organización o consorcios, lo que puede ser una opción viable en algunos casos.

En cuanto a la privacidad y confidencialidad de los datos, los oráculos que interactúan con fuentes de datos externas deben ser evaluados cuidadosamente, independientemente de si la BC es pública o privada. Esto es especialmente relevante en el contexto de la gestión de SLA, donde podrían estar involucrados datos sensibles o confidenciales.

De forma general algunas consideraciones clave para la elección de un oráculo son:

- Confiabilidad y precisión de los datos: El oráculo debe proporcionar datos precisos y confiables del mundo real, ya que estos son utilizados para evaluar el cumplimiento de los SLA.

- Descentralización y resistencia a la manipulación: Es recomendable optar por oráculos descentralizados que minimicen los riesgos de manipulación de datos o puntos únicos de falla.

- Escalabilidad y rendimiento: El oráculo debe ser capaz de manejar un gran volumen de solicitudes y proporcionar datos en tiempo real, dado que los SLA suelen tener requisitos estrictos de monitoreo y cumplimiento.

- Costos y tarifas: Es importante considerar los costos asociados al uso del oráculo, como tarifas de transacción, suscripciones o pagos por solicitud de datos.

- Soporte y documentación: Elegir un oráculo con una comunidad activa, buena documentación y soporte técnico puede facilitar la integración y el mantenimiento a largo plazo.

3.9 Ejemplo de implementación

En la sección 3.1 se planteó de forma general una propuesta de esquema de gestión de SLA que utiliza BC y SC. El esquema anterior sirve de base para construir el diagrama de la Figura 23. En este diagrama se establece una correspondencia con tecnologías específicas para cada componente del esquema propuesto y constituye un ejemplo de implementación práctica de dicho esquema. Se observa en el diagrama que tanto cliente como proveedor tienen un monedero digital que les permite realizar solicitudes con el fin de ejecutar acciones en la BC por medio de la dApp. Como respuesta, la dApp retorna notificaciones del resultado de su solicitud. La dApp, para este caso, se despliega en AWS EC2, un servicio de la empresa Amazon que proporciona acceso a máquinas virtuales como servicio en la nube de Amazon.

En una máquina virtual AWS EC2 se ejecuta el *frontend* y el *backend* de la aplicación distribuida. El *frontend* es escrito usando el *framework* de JavaScript llamado React.js, el cual proporciona una interfaz de usuario para la interacción de cliente y proveedor con el esquema propuesto. El *backend* de la dApp se escribe utilizando Node.js en combinación con el *framework* Express.js para asumir la administración del almacenamiento distribuido en la red IPFS. Se utiliza además para el *backend* la librería web3.js para ejecutar acciones en la BC de forma programática según las solicitudes de los usuarios.

Se utiliza Ethereum como BC de tipo pública. En el diagrama Ethereum ofrece soporte a la ejecución de los SC que son codificados en *Solidity*. SC Mercado es el orquestador principal para la creación de los SLA y gestión de las subastas de SLA, y es el primer SC que se requiere desplegar para la puesta en marcha del esquema. SC Subasta gestiona la subasta de un SLA en específico. SC SLA es responsable de la ejecución del SLA, por lo que verifica si se cumplen los umbrales establecidos según los parámetros de QoS plasmados en el SLA correspondiente. La función de SC SLA encargada de la solicitud de KPI se ejecuta de forma periódica por un elemento externo a la BC, que para este caso es la funcionalidad *Chainlink Keepers* del Oráculo distribuido *Chainlink*. La función ejecutada periódicamente realiza una solicitud a la funcionalidad *Chainlink ANY API*, para obtener los datos de monitoreo de la API del proveedor, que es la

NEF de la red 5G. Estas métricas se utilizan para definir las violaciones o incidencias en el servicio y quedan registradas de forma inmutable en la BC Ethereum. Tanto las métricas como las violaciones son visibles por el cliente en la dApp y en la BC Ethereum de forma pública.

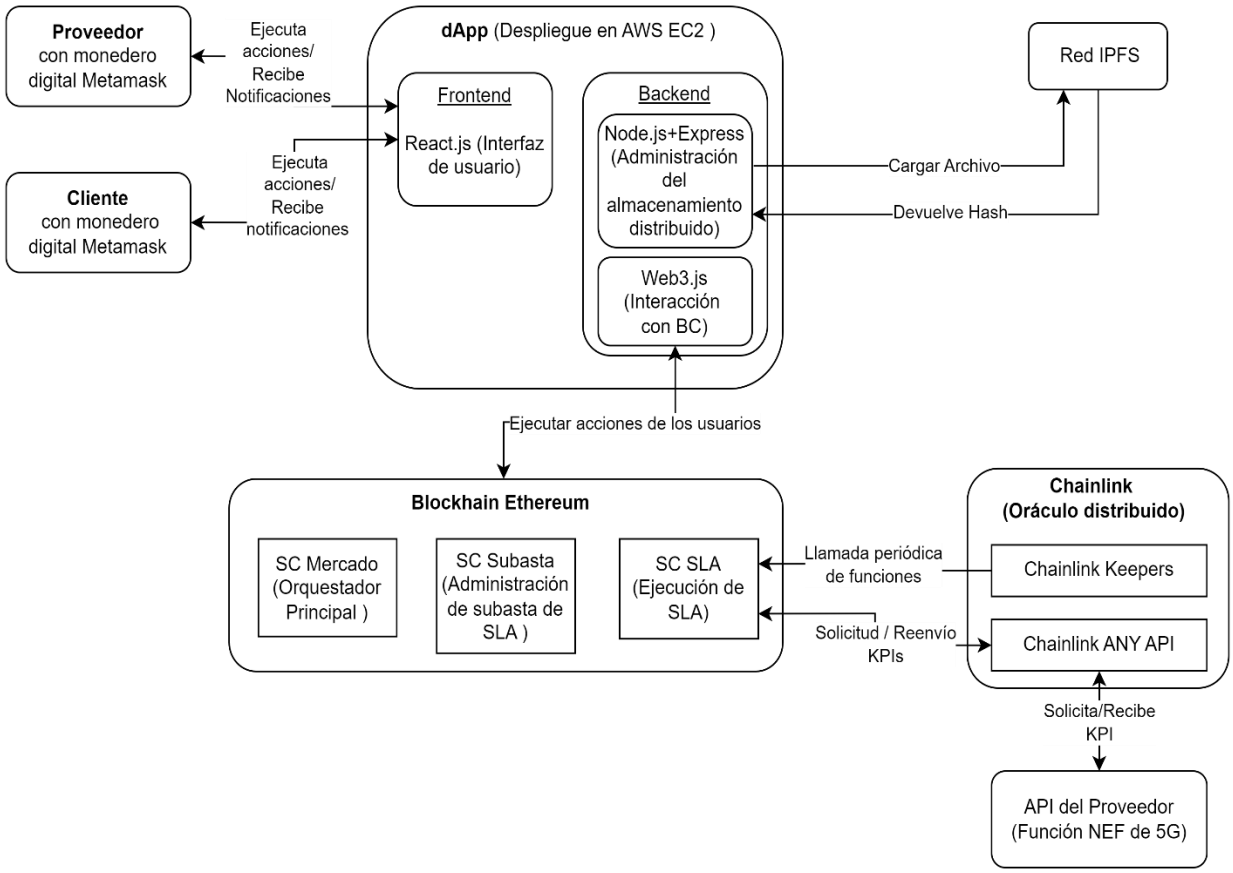

Figura 23. Ejemplo de implementación de esquema propuesto [Elaboración Propia].

3.10 Conclusiones

En este capítulo, se ha propuesto un esquema integral para la gestión automatizada de SLA de NS en entornos 5G/B5G, se han discutido los componentes constituyentes del esquema, las interacciones entre ellos y se han recomendado tecnologías para la implementación de varios componentes del esquema.

Se puede concluir lo siguiente:

- Se presentó el diseño de un esquema para la gestión de SLA de NS que aprovecha la tecnología Blockchain y los contratos inteligentes para brindar una gestión transparente, confiable y descentralizada de los SLA para segmentos de red en entornos 5G/B5G.

- La arquitectura propuesta incluye componentes clave como una dApp, la red IPFS, contratos inteligentes, un oráculo distribuido y una API expuesta por los proveedores para acceder a los datos de monitoreo de los NS.

- Se describe a detalle cada uno de estos componentes, así como los flujos y procedimientos en los que se interrelacionan. A través de diagramas de secuencia se muestra el flujo y se establece la lógica de la operación del esquema diseñado.

- El esquema propuesto permite una gestión automatizada de SLA de NS donde la carga de trabajo para los proveedores sea más reducida y los clientes tengan mayor confiabilidad en comparación a las soluciones tradicionales, uno de los objetivos a cumplir.

- La utilización de contratos inteligentes para codificar las reglas y condiciones para la correcta gestión de los SLA en cada una de sus fases posibilita una ejecución confiable de los acuerdos sin la necesidad de una autoridad centralizada.

- Se abordaron recomendaciones y consideraciones clave para la implementación del esquema, incluyendo la selección de tecnologías para el despliegue de la dApp, para la ejecución de SC, para la conexión con una BC, así como para el oráculo distribuido.

Capítulo 4. Validación del sistema propuesto

En este capítulo se presenta una prueba de concepto para el esquema de gestión de SLA de NS propuesto. Se persigue validar a partir de una implementación del esquema la eficacia en la gestión de SLA y comprobar la verificación de su cumplimiento en un entorno BC local. Otro objetivo de este capítulo es analizar el desempeño del esquema según el uso de las BC de prueba *Ethereum Sepolia* y *Polygon Mumbai*, de forma que se muestre el impacto en cuanto a tiempo y costos en la utilización de diferentes BC para el esquema. De igual forma, se realiza otra prueba en las *testnet,* con el objetivo de comprobar la viabilidad del uso del oráculo *Chainlink* para la monitorización en entornos de SLA de NS en un periodo común de medición de 1 minuto.

En este capítulo se presenta un procedimiento de validación del esquema propuesto, que guiará la realización de las pruebas. Se plantea la configuración del esquema como escenario para la validación, así como el conjunto de herramientas para llevarla a cabo. Se presentan pruebas locales y pruebas en *testnet* obteniéndose resultados que son discutidos y analizados a profundidad.

4.1 Procedimiento de validación

Para asegurar el cumplimiento de los requisitos establecidos, es crucial construir una metodología sólida del proceso de validación del esquema. La metodología proporciona una guía estructurada que garantiza la repetibilidad del proceso de validación, el seguimiento de buenas prácticas y estándares y la consistencia en los resultados a obtener. Por esta razón, se estableció una metodología para realizar una prueba de concepto, cuya finalidad es implementar el esquema de gestión de SLA de NS propuesto en una red de prueba. Las fases de la metodología se muestran a continuación:

1. Selección de tecnologías y herramientas: Comprende la selección de las tecnologías a utilizar en cada componente y las tecnologías necesarias para llevar a cabo la codificación y desarrollo

2. Desarrollo de pruebas locales para SC: Consiste en codificar los contratos inteligentes que implementarán la lógica del esquema, probar exhaustivamente los contratos mediante pruebas unitarias y de integración para verificar su correcta operación en una BC local.

3. Programar la lógica del lado de los usuarios: Aquí se debe integrar la interacción con los contratos inteligentes por medio del uso de bibliotecas o SDK de forma que se implementen posibles flujos de operación de los usuarios en una *testnet*.

4. Pruebas y Despliegue en Redes de Prueba: En esta fase se despliegan los SC en una red de prueba (*testnet*), posteriormente se analizan costos y tiempos de ejecución de las funciones de los SC que intervienen en las acciones que pueden ejecutar proveedor y cliente en la BC.

4.2 Tecnologías y herramientas

En el capítulo 3 se detallaron opciones de tecnologías para varios de los componentes del esquema y los criterios de elección para una u otra. Estos criterios se utilizaron para configurar los dos escenarios que se muestran en la Figura 24 y en la Figura 25, sobre los que se realizaron simulaciones y pruebas (*test*). En nuestra prueba de concepto se selecciona la red IPFS como sistema distribuido para el almacenamiento de los documentos SLA en forma de archivo. Se utilizarán *testnet* de las BC públicas, *Ethereum* primero (Figura 24) y en segundo lugar *Polygon* (Figura 25), ambas basadas en la máquina virtual de *Ethereum* (EVM, por las siglas del término en inglés*, Ethereum Virtual Machine*), que es una tecnología de máquina virtual que ejecuta los contratos inteligentes en un entorno BC.

El oráculo distribuido utilizado es *Chainlink* y para la API del proveedor se construye una abstracción de esta API, se simula con una API creada a partir de un *framework* del lenguaje de programación *Python* llamado *Flask*. La API responde a la solicitud de datos de monitorización.

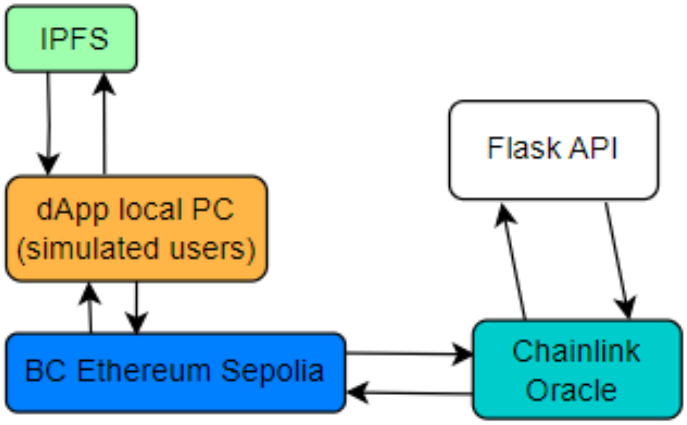

Figura 24. Tecnologías para primer escenario de simulación [Elaboración propia].

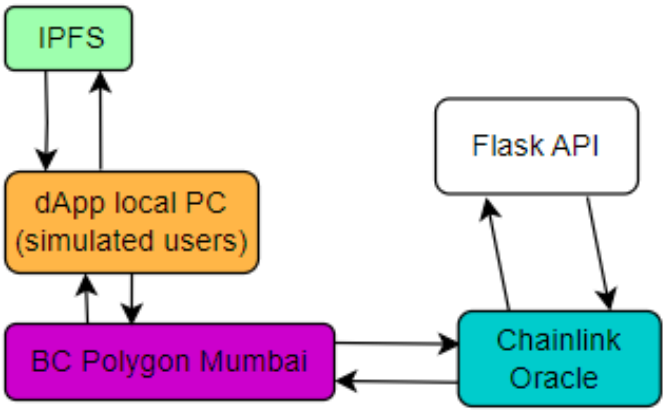

Figura 25. Tecnologías para segundo escenario de simulación [Elaboración propia].

Es importante destacar que *Flask* (Flask, 2024) es un *framework* Web utilizado ampliamente para el desarrollo de aplicaciones Web y API. Su elección para implementar la abstracción de la API del proveedor se debe a que es ligero y fácil de configurar, lo que lo convierte en una opción ideal para crear API simples y eficientes. Además, ofrece una gran flexibilidad y modularidad, permite agregar y personalizar funcionalidades según las necesidades del proyecto. *Flask* también cuenta con una amplia comunidad y una documentación sólida. *Flask*, en general, es una elección atractiva para implementar API debido a su simplicidad, flexibilidad, integración con herramientas existentes y su enfoque en el rendimiento y la escalabilidad.

Para llevar a cabo la validación de forma rápida y eficiente, es fundamental contar con un conjunto sólido de herramientas y tecnologías especializadas para el desarrollo de SC. Estas herramientas deben proporcionar un entorno de desarrollo integral que facilite cada etapa del proceso, desde la codificación, la implementación y el mantenimiento. En este sentido se requiere elegir un lenguaje de programación de SC, un editor de código, alguna herramienta de control de versiones para el proyecto y un espacio de trabajo (*framework*) que facilite la ejecución de pruebas.

4.2.1 Lenguaje de programación de SC

Como lenguaje de programación para los SC se utilizó *Solidity* (Solidity, 2024). Como se puede observar en las Tablas 3 y 4 del capítulo 3, es el lenguaje de programación de SC más utilizado, tanto en BC privadas como públicas. Esto se debe a que *Solidity* fue específicamente diseñado para la EVM, varias BC tienen su

base en esta tecnología. A continuación, se presentan las ventajas que tiene *Solidity* con respecto a otros lenguajes de programación:

- Madurez y adopción: *Solidity* es el lenguaje de SC más maduro y ampliamente adoptado. Cuenta con una gran comunidad de desarrolladores, abundantes recursos de aprendizaje, herramientas y bibliotecas de código abierto bien establecidas.

- Facilidad de auditoría y análisis: El código de *Solidity* es relativamente legible y sigue una estructura similar a otros lenguajes de programación. Esto facilita la auditoría y el análisis de contratos inteligentes, aspectos cruciales para garantizar la seguridad y la corrección de los contratos antes de su implementación.

- Herramientas y *frameworks* de desarrollo maduros: *Solidity* cuenta con un ecosistema de herramientas y *frameworks* de desarrollo maduros, como *Truffle, Remix, Hardhat*, entre otros (Dominguez & Monzon Baeza, 2024). Estas herramientas facilitan el desarrollo, la prueba, el despliegue y la interacción con los contratos inteligentes.

- Soporte para bibliotecas y herencia: *Solidity* admite el uso de bibliotecas reutilizables y la herencia de contratos, lo que facilita la modularidad, la reutilización de código, y promueve un desarrollo más eficiente y mantenible.

- Verificación formal: *Solidity* tiene soporte para la verificación formal de SC, lo que permite detectar errores y vulnerabilidades de manera rigurosa antes de la implementación, de forma que mejora la seguridad y la confiabilidad de los contratos.

La estructura de un SC escrito en *Solidity* de forma general se observa en la Figura 26. Normalmente se comienza con un identificador de licencia, seguido por una línea de código que especifica la versión del compilador a utilizar. A continuación, se pueden incluir importaciones de otros contratos o bibliotecas necesarias. El cuerpo principal del contrato se inicia con la palabra clave *contract* seguida del nombre del contrato.

En el cuerpo del contrato se declaran las variables de estado que almacenan la información persistente del contrato. Los eventos se definen para emitir notificaciones, mientras que los errores personalizados permiten manejar situaciones específicas. El constructor es una función especial que se ejecuta una sola

ocasión al desplegar el contrato, se utiliza para inicializar el estado. Las funciones del contrato constituyen el núcleo de su funcionalidad, pueden ser públicas, privadas, internas o de solo lectura (*view/pure*). Estas funciones aceptan modificadores para añadir comportamientos comunes o restricciones, por ejemplo, para que solo determinados usuarios puedan llamarla. Esta organización proporciona una estructura clara, y modular para el desarrollo de contratos inteligentes, y facilita su implementación, legibilidad, así como, el mantenimiento en el ecosistema de la BC.

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

// Importar otros contratos aquí si es necesario

contract MiContrato {
    // Estado del contrato
    uint256 public miVariable;

    // Evento que se emite cuando se actualiza miVariable
    event MiEvento(uint256 newValue);

    // Error personalizado
    error ValorDemasiadoAlto(uint256 valor);

    // Constructor que se ejecuta una sola vez al ser desplegado el contrato
    constructor(uint256 initialValue) {
        miVariable = initialValue;
    }

    // Función para actualizar el valor de miVariable
    function actualizarVariable(uint256 newValue) public {
        if (newValue > 100) {
            revert ValorDemasiadoAlto(newValue);
        }
        miVariable = newValue;
        emit MiEvento(newValue);
    }

    // Función de ejemplo que devuelve el valor actual de miVariable
    function obtenerVariable() public view returns (uint256) {
        return miVariable;
    }
}
```

Figura 26. Estructura general de un SC escrito en *Solidity* [Elaboración Propia].

4.2.2 Editor de código

En el ámbito de la programación de SC, *Visual Studio Code* ha emergido como una herramienta de desarrollo de código abierto altamente versátil y potente. Su arquitectura modular y extensible lo convierte en una opción atractiva para los desarrolladores que buscan un entorno de desarrollo integrado personalizable y adaptado a sus necesidades específicas. Uno de sus aspectos más destacados es su capacidad para integrar una amplia gama de extensiones y complementos. Estas extensiones permiten ampliar las funcionalidades nativas del editor, y proporcionan características adicionales que mejoran la experiencia de desarrollo de contratos inteligentes. Por estas razones, se seleccionó para la codificación de los SC en la prueba de concepto.

Entre las extensiones que tiene *Visual Studio Code* para el desarrollo de SC esta *Solidity*, que tiene el mismo nombre que el lenguaje de programación a utilizar. Esta extensión permite: (Visual Studio Code, 2024):

- Completar código de forma automática,
- Atajos para dirigirse a referencias, definición y definición de tipo,
- Dar formato al código Solidity,
- Validación de código en línea a partir de errores/advertencias del compilador para proyectos de *Hardhat* o *Foundry*,
- Ayuda emergente para variables, llamadas a funciones, errores, eventos, etc.,
- Acciones de código como soluciones rápidas sugeridas a partir de errores o advertencias del compilador para proyectos de Hardhat o Foundry.

Estas características, combinadas con la capacidad de personalización y extensibilidad de *Visual Studio Code*, crean un entorno de desarrollo altamente productivo y eficiente para el desarrollo de SC. Al minimizar la posibilidad de errores sintácticos y de formato, viabilizan una codificación más fluida y un entendimiento más profundo del código. Como se aprovechan las herramientas integradas de autocompletado, navegación y refactorización al realizar la codificación es posible centrarse en la lógica y la funcionalidad de los contratos, y optimizar el tiempo y los recursos dedicados al desarrollo.

4.2.3 Herramienta para control de versiones

Para el control de versiones en el desarrollo de este proyecto se usará la *Git* como herramienta. Esta herramienta ofrece varias ventajas importantes entre las cuales se encuentran (Git, 2024):

- Historial de cambios: Git mantiene un registro detallado de todos los cambios realizados en el código, lo que facilita el seguimiento y la auditoría de las modificaciones a lo largo del tiempo.
- Ramificación y experimentación: Gracias a las ramas, Git permite crear líneas de desarrollo independientes para experimentar con nuevas funcionalidades sin afectar la rama principal.

- Reversibilidad: Si se detectan errores o vulnerabilidades, Git permite revertir el código a una versión anterior estable y confiable.

- Revisión y pruebas: Git facilita la revisión de código y la ejecución de pruebas antes de fusionar los cambios, asegura la calidad y la seguridad de los contratos inteligentes.

Git brinda control, colaboración, seguridad, calidad y eficiencia en el desarrollo de contratos inteligentes, convirtiéndose en una herramienta indispensable para gestionar cada fase en la codificación en un proyecto.

4.2.4 Framework de SC.

Para la codificación de los SC del esquema se utilizó *Solidity*, como se comentó anteriormente, pero bajo el marco de trabajo *Foundry* (Foundry, 2024) principalmente porque es más adecuado para proyectos que valoran la velocidad, la eficiencia y la capacidad de escribir pruebas directamente en *Solidity*. El uso de este *framework* tiene como requisito usar *Linux* para el desarrollo, y tiene 3 herramientas que lo convierten en una solución excelente para el desarrollo y auditoría de SC. Estas herramientas son:

- *Cast*: Es una interfaz de consola de comandos que permite interactuar con contratos inteligentes previamente desplegados, enviar transacciones y leer datos de la red BC configurada

- *Anvil*: Una BC local con una implementación simple

- *Forge*: Una herramienta para la compilación de SC, ejecución de script y test.

4.3 Pruebas locales

Durante la fase de desarrollo de pruebas locales se llevó a cabo un enfoque integral donde la codificación de los SC se realizó en paralelo con la construcción de pruebas para comprobar el correcto flujo de ejecución de los SC, lo cual asegura su funcionalidad y comportamiento esperado en un entorno local controlado. La implementación de pruebas locales desempeña un papel crucial al brindar la capacidad de validar cada aspecto de la lógica de los contratos, identificar posibles errores o inconsistencias de manera

temprana, y garantizar la confiabilidad del código antes de su despliegue en una *testnet* o una red BC real.

Las pruebas locales se llevaron a cabo en la BC *Anvil* la cual es una implementación completa de una BC *Ethereum* que permite un inicio casi instantáneo del entorno de desarrollo y pruebas de forma local. No depende de servicios externos ni requiere sincronización, lo que la hace ideal para un entorno de desarrollo y pruebas aislado. Al utilizar esta implementación se tiene un escenario más controlado, ya que se pueden configurar aspectos como el saldo inicial de las cuentas, el código de bytes preinstalado y el comportamiento de los métodos de prueba. En su caso las transacciones enviadas a la BC local se minan automáticamente, lo que facilita las pruebas y la interacción con los SC.

Utilizando el lenguaje de programación *Solidity* existen varias acciones que se pueden realizar como parte de un *test* para verificar el comportamiento y la funcionalidad de los contratos inteligentes de forma automatizada. Algunas de las acciones más comunes que se pueden realizar en un *test* escrito en *Solidity* incluyen:

- Despliegue de contratos: Los *tests* pueden incluir el despliegue de uno o varios contratos inteligentes en la BC configurada (local, *testnet* o real).

- Llamadas a funciones: Los *tests* pueden realizar llamadas a las funciones públicas de los contratos desplegados, comparten los parámetros necesarios y verifican los resultados esperados.

- Verificación de eventos: Después de interactuar con los contratos, los *tests* pueden verificar si se emitieron los eventos correspondientes con los datos correctos.

- Verificación del estado del contrato: Los *tests* pueden comprobar el estado interno de los contratos, como el valor de las variables de estado, el saldo de cuentas o el contenido del almacenamiento.

- Simulación de transacciones: Es posible simular la ejecución de transacciones en la BC local, verificar su éxito o fracaso, así como los cambios resultantes en el estado de los contratos. En este caso también se puede definir quién es el que hace la transacción y cuál es el destino de la misma.

- Pruebas de revertido: Los *tests* pueden intentar provocar situaciones de revertido o fallo en la ejecución de los contratos, y aseguran el manejo correcto de estos casos.

- Pruebas de acceso y permisos: Se pueden realizar pruebas para verificar que solo las cuentas autorizadas puedan ejecutar ciertas funciones o acceder a ciertos datos del contrato.

Se utilizan combinaciones de estas acciones para construir una suite completa de *test*, tanto unitarios como de integración. Para comprobar que la interacción con la BC está implementada correctamente en los SC Mercado, SC SLA y SC Subasta, según los flujos definidos en el esquema, se implementaron varios flujos de prueba en scripts considerando *Anvil* como BC local.

Es importante destacar la diferencia entre los tipos de test. Un test unitario de SC verifica el comportamiento correcto de una función o componente específico del contrato de forma aislada, y asegura que cada parte del SC específico funcione según lo esperado. Por otro lado, un test de integración evalúa cómo interactúan múltiples contratos o componentes entre sí con el entorno de la BC, verificando que el sistema completo de contratos inteligentes funcione adecuadamente como un todo y maneje correctamente las transacciones y el estado de la BC.

En la Figura 27 se expone un ejemplo en Solidity de un test unitario. El ejemplo muestra como en un SC tipo test se pueden configurar direcciones para clientes o proveedores y asignar valores a las variables de estado del contrato. Se puede observar que se puede hacer un despliegue local del SC Subasta y luego establecer balances iniciales para los clientes como ocurre en la función setUp(). La función testBidLowerThanHighestBidGetReverted() constituye una prueba unitaria. En un contrato tipo test la función se identifica como prueba cuando empieza con el prefijo “test”. Su tarea consiste en comprobar si la oferta por parte del CLIENTE 1 en una subasta de SLA no es aceptada porque es menor que el valor mínimo de oferta establecido en el SLA, que para este caso se fijó 0.1 ether (criptomoneda de Ethereum) en las variables de estado del contrato.

Como se observa en la Figura 28, al ejecutar el test con la herramienta forge del framework Foundry, se obtiene que el SC Subasta (Auction.sol) pasa la prueba sin errores. El resultado de esta prueba demuestra la efectividad de un test específico diseñado para verificar el comportamiento correcto de la subasta de SLA. La ejecución exitosa del test ([PASS]) indica que el SC Subasta manejó adecuadamente este escenario, revirtiendo la transacción como se esperaba.

La rápida ejecución en 22.57 ms facilita la realización frecuente de pruebas durante el desarrollo. Al ejecutar solo este test específico, se permite un enfoque preciso en esta funcionalidad particular del contrato de subasta.

```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

import {Test, console} from "forge-std/Test.sol";
import {DeployAuction} from "../script/DeployAuction.s.sol";
import {Auction} from "../src/Auction.sol";

contract AuctionTest is Test {
    Auction public auction;
    address public CLIENT_1 = makeAddr("client1");
    address public CLIENT_2 = makeAddr("client2");
    address public CLIENT_3 = makeAddr("client3");
    uint256 public biddingTime = 1 days;
    uint256 public startValue = 0.01 ether;
    uint256 public constant STARTING_BALANCE = 100 ether;

    function setUp() external {
        DeployAuction deployer = new DeployAuction();
        auction = deployer.run();
        vm.deal(CLIENT_1, STARTING_BALANCE);
        vm.deal(CLIENT_2, STARTING_BALANCE);
        vm.deal(CLIENT_3, STARTING_BALANCE);
    }

    function testBidLowerThanHighestBidGetReverted() public {
        vm.prank(CLIENT_1);
        uint256 bidAmount = 0.001 ether;
        vm.expectRevert();
        auction.bid{value: bidAmount}();
    }
```

Figura 27. Ejemplo de *test* unitario [Elaboración Propia].

```
(myvenv) ares9505@Ariel:~/foundry/foundry-SLAuto$ forge test --match-test testBidLowerThanHighestBidGetReverted
[⠒] Compiling...
No files changed, compilation skipped

Ran 1 test for test/Auction.t.sol:AuctionTest
[PASS] testBidLowerThanHighestBidGetReverted() (gas: 21770)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 22.57ms (139.78µs CPU time)
```

Figura 28. Respuesta del *test* unitario de ejemplo [Elaboración Propia].

En la Figura 29 se expone un ejemplo en Solidity de un test de integración. Para este ejemplo el test de integración verifica el comportamiento correcto de la creación de un SC SLA y su estado inicial de activación. Primero obtiene la dirección del propietario del SC Mercado y simula que las siguientes acciones se realicen a nombre del propietario con la instrucción vm.prank(owner). Solo un proveedor puede crear un SLA, y el creador de SC Mercado es el proveedor que se añade por defecto. La acción siguiente es, crear un nuevo contrato SLA llamando a createSLA() y obtener su dirección. El test procede a verificar el estado de activación del SLA recién creado llamando a getSlaActivationState(). Finalmente,

comprueba que el SLA esté inactivo tras su creación. Este test es crucial para asegurar que el proceso de creación de SLA funciona correctamente y que los contratos comienzan en el estado apropiado, y demuestra la correcta integración entre el contrato de mercado y el contrato SLA en un contexto más amplio del sistema.

```
function testSLAinactiveBeforeCreation() public {
    address owner = market.getOwner();
    vm.prank(owner);
    (address slaAddress, ) = createSLA();

    bool activationState = SLA(payable(slaAddress)).getSlaActivationState();
    assert(activationState == false);
}
```

Figura 29. Ejemplo de *test* de integración escrito en *Solidity* [Elaboración Propia].

4.3.1 Resultados de pruebas locales

Con respecto a las pruebas locales se obtienen como resultado el porcentaje de SC cubierto con las pruebas y la efectividad en el cumplimiento de los flujos del esquema como se verá posteriormente mediante la utilización de la herramienta *forge* .

File	% Lines	% Statements	% Branches	% Funcs
src/Auction.sol	90.32% (28/31)	90.91% (30/33)	75.00% (15/20)	100.00% (6/6)
src/Market.sol	76.92% (10/13)	76.47% (13/17)	100.00% (0/0)	60.00% (3/5)
src/SLA.sol	90.36% (75/83)	88.99% (97/109)	71.88% (23/32)	71.43% (15/21)

Figura 30. Resultado de la prueba de cobertura de SC [Elaboración Propia]

Al utilizar el comando *forge coverage* se puede medir el porcentaje probado para cada *Smart Contract*, lo que es crucial para evaluar la cobertura de pruebas y la fiabilidad del código. Con este comando se pueden identificar áreas críticas que requieren una mayor cobertura de pruebas. La Figura 30, muestra los resultados de la ejecución de *forge coverage* para los contratos inteligentes SLA, Mercado y Subasta.

El informe generado presenta columnas que proporcionan información detallada sobre la cobertura de las pruebas:

- *File*: Identifica el contrato inteligente analizado.

- *% Lines*: Indica el porcentaje de líneas de código ejecutadas durante las pruebas en relación con el total de líneas del contrato.

- *% Statements*: Representa el porcentaje de declaraciones ejecutadas. Aunque similar al porcentaje de líneas, esta métrica difiere ya que una sola línea puede contener múltiples declaraciones.

- *% Branches*: Muestra el porcentaje de ramas de código ejecutadas. Estas ramas suelen originarse de estructuras condicionales en el contrato.

- *% Funcs*: Revela el porcentaje de funciones invocadas durante la ejecución de las pruebas.

En total se construyeron 32 test, 14 unitarios y el resto de integración. La herramienta forge nos da la opción de correr todos los test de forma simultánea como se observa en la Figura 31, donde todos los SC en una BC local bajo prueba pasan los test.

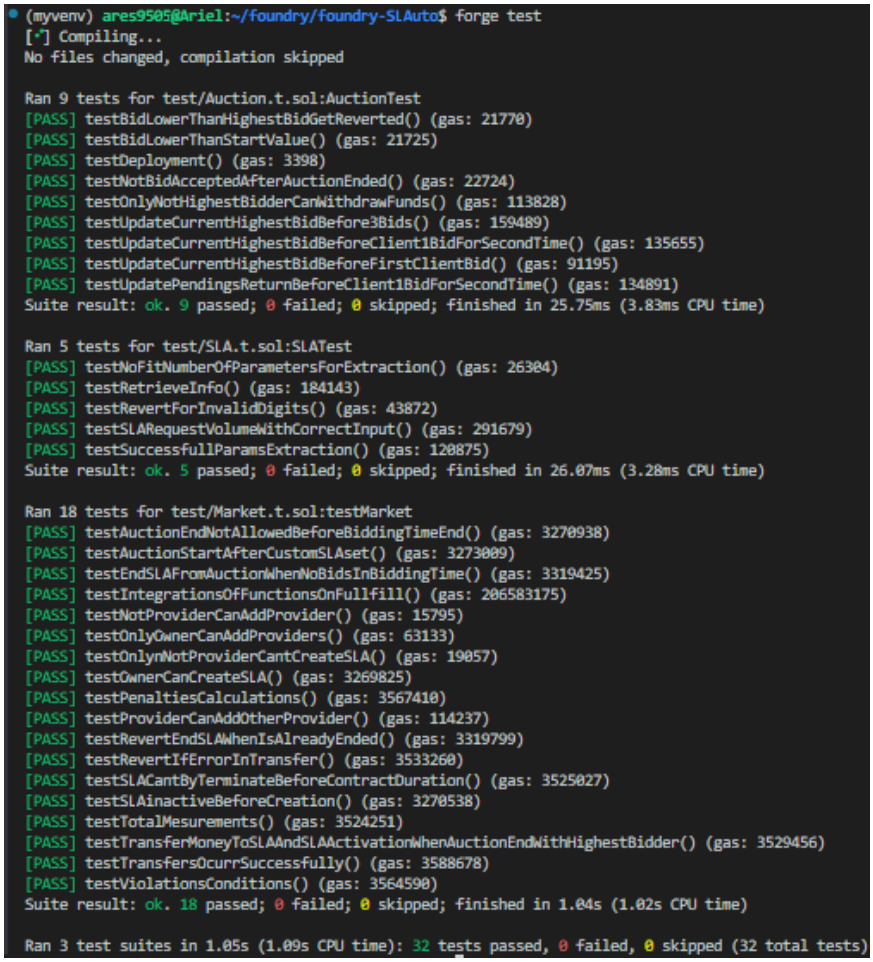

Figura 31. Resultados de *test* en BC local [Elaboración Propia].

4.4 Pruebas en *testnet*

Para la realización de pruebas en *testnet,* es fundamental definir la ubicación y configuración de cada tecnología elegida, integrándolas en una arquitectura que representa el esquema propuesto. La Figura 32 ilustra el escenario de las pruebas en *testnet,* detallando la disposición de los componentes en correspondencia al esquema de la Figura 13.

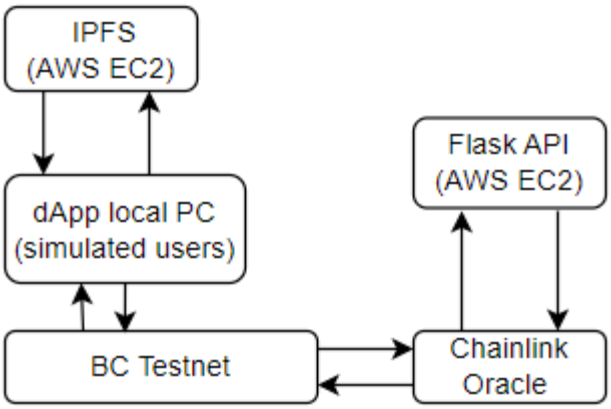

Figura 32. Configuración del esquema propuesto para pruebas en *testnet* [Elaboración Propia].

El núcleo del entorno de pruebas se centra en un PC local con 8GB de memoria RAM y un procesador Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz. En este equipo, se simulan las interacciones de la dApp mediante el uso de Web3.py, un SDK escrito en *Python* que facilita la comunicación con redes BC.

Para optimizar el rendimiento y la conectividad, se implementan varios componentes en la nube. Un cliente IPFS configurado en una máquina virtual EC2 de AWS, permite la conexión eficiente con la red IPFS desde la PC local, y asegura un almacenamiento descentralizado. Se seleccionaron dos *testnet* públicas compatibles con la tecnología Ethereum, permitiendo la validación del funcionamiento en múltiples entornos. Estas son Ethereum Sepolia que funciona con mecanismo de consensos PoS y *Polygon Mumbai* cuyo mecanismo de consenso es PoA.

El oráculo *Chainlink*, que opera sobre internet, proporciona datos externos confiables a los contratos inteligentes desde la API simulada del proveedor la cual se configuró en otra máquina virtual EC2 en AWS. Esta API simula la funcionalidad del proveedor real, y suministra diversas métricas de monitoreo esenciales para las pruebas, las cuales son:

- Latencia: El tiempo que tarda un paquete de datos en viajar desde su origen hasta su destino. Se mide generalmente en milisegundos.

- *Throughput:* La cantidad de datos que se pueden transmitir en una unidad de tiempo. Se expresa comúnmente en bits por segundo (bps).

- *Jitter*: La variación en el tiempo de llegada de los paquetes. Un *jitter* alto puede causar problemas en aplicaciones en tiempo real como videollamadas.

- Ancho de banda digital: La capacidad máxima teórica de transmisión de datos de una red. Se mide en bits por segundo (bps).

- Pérdida de paquetes: El porcentaje de paquetes que no llegan a su destino. Puede afectar significativamente la calidad del servicio.

- Confiabilidad del servicio: La medida de la consistencia de la red para proporcionar el servicio esperado sin fallos o interrupciones.

- Tiempo de supervivencia: El período durante el cual un servicio o sistema permanece operativo sin interrupciones.

- Tiempo de interrupción: La duración total durante la cual un servicio o sistema no está disponible o funcionando correctamente.

Un aspecto importante de esta prueba es que, a diferencia de experimentos y pruebas de trabajos anteriores, tienen en cuenta varias métricas para la monitorización. Es importante incluirlas en las pruebas pues son más parámetros que debe analizar el SC, ya que el volumen de las métricas puede impactar negativamente en el desempeño del esquema.

Para asegurar una conexión con la BC objetivo, se aprovechó la capa gratuita ofrecida por *Alchemy*, un proveedor líder de infraestructura BC como se recomienda en la sección 3.8.2. *Alchemy* proporciona nodos gestionados de alta disponibilidad con lo que elimina la necesidad de mantener y sincronizar nodos propios, lo que simplifica significativamente el proceso de desarrollo y pruebas.

Desde la PC local, donde reside la simulación de la dApp se configuró la comunicación con la API de *Alchemy* que implica la integración de URL RPC en el código de la aplicación. *Alchemy* actúa como un intermediario eficiente, y facilita una interacción fluida y segura con las varias BC públicas como se observa en la Figura 33.

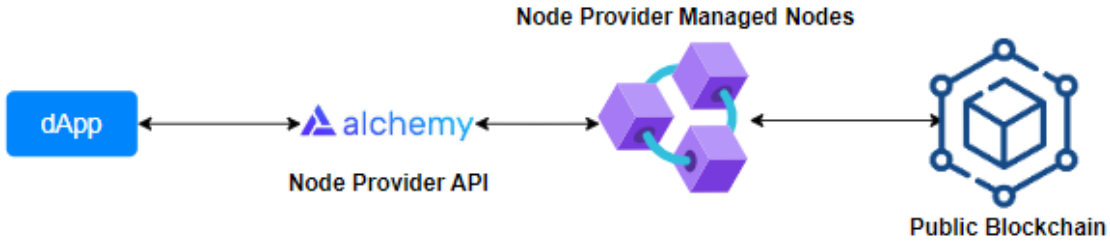

Figura 33. Conexión de la dApp a la BC pública por medio de un proveedor de nodos [Elaboración propia].

La infraestructura de *Alchemy* puede manejar un volumen alto de solicitudes, crucial para simular condiciones de carga realistas. Los nodos gestionados de *Alchemy* garantizan una alta disponibilidad y reducen los tiempos de inactividad. Al utilizar nodos en la nube, se libera capacidad de procesamiento local para otras tareas de desarrollo. Además, *Alchemy* proporciona herramientas de análisis que permiten supervisar el rendimiento de las interacciones con la BC. *Alchemy* facilita la conexión con diferentes redes BC y permite pruebas en diversos entornos sin cambios significativos en la configuración.

Esta configuración no solo simplifica el proceso de desarrollo y pruebas, sino que también emula de manera más precisa las condiciones de un entorno de producción y proporciona una base sólida para la validación y optimización de la dApp antes de su despliegue en la red principal.

Un requisito para simular y realizar pruebas de operación es considerar la tecnología de monedero digital para los actores simulados del esquema, dígase proveedores y clientes. Cada monedero digital almacena la dirección pública y su clave privada asociada, necesaria para que los actores puedan firmar transacciones e interactuar con el esquema.

En trabajos como (Saian et al., 2024) se recomienda la tecnología *Metamask* como monedero digital. A continuación, se muestran ventajas de *Metamask* que justifican su elección como monedero digital para la validación del esquema (Gonzalez et al., 2021) :

- *Metamask* se distingue por su interfaz intuitiva y amigable, lo que facilita su uso e interacción, y brinda una experiencia fluida y accesible en la realización de pruebas a la dApp.

- Al ser un software de código abierto, se beneficia de una comunidad comprometida de desarrolladores que trabajan incansablemente para mejorar y actualizar continuamente la plataforma. Esto garantiza que los usuarios tengan acceso a las últimas funcionalidades, parches de seguridad y optimizaciones.

- Se destaca por su portabilidad y facilidad de instalación. No es necesario descargar ni instalar software adicional en el ordenador local. Es compatible con una amplia gama de navegadores populares, incluyendo Firefox, Chrome, Opera y Brave, basta con agregar la extensión al navegador Web preferido.

Otra función importante de un monedero digital es albergar criptomonedas que permiten hacer las transacciones. En el caso de Metamask admite criptomonedas compatibles con la tecnología Ethereum, aunque tiene cierta flexibilidad e interoperabilidad para el trabajo con criptomonedas de otro tipo de redes BC.

En los entornos de BC públicas reales, las criptomonedas funcionan como activos financieros digitales con valor económico tangible, facilitan transacciones y operaciones financieras descentralizadas. Estas criptomonedas, se negocian en mercados globales y pueden intercambiarse por monedas fiduciarias.

En contraste, las redes de prueba o *testnet* utilizan versiones simuladas de estas criptomonedas. Estas monedas de prueba, aunque técnicamente idénticas a sus contrapartes reales, carecen de valor financiero en el mundo real. Su propósito principal es emular el comportamiento y las características de las criptomonedas reales en un entorno controlado y sin riesgo.

Las criptomonedas de prueba, comúnmente conocidas como *testnet coins* o *faucet coins*, se distribuyen gratuitamente a través de servicios Web especializados llamados *faucets*. Los *faucets* son plataformas diseñadas para dispensar pequeñas cantidades de estas monedas de prueba a desarrolladores, investigadores y entusiastas de la tecnología BC.

La utilización de *faucets* o monedas de prueba ofrece múltiples ventajas que se aprovechan en este trabajo. Permiten experimentar con contratos inteligentes y aplicaciones dApp sin riesgo financiero. Proporcionan un entorno que replica fielmente las condiciones de la red principal, incluyendo gastos de transacción y limitaciones de recursos. Permite identificar y corregir errores en el código antes del

despliegue en la red principal y posibilita la verificación de la correcta integración con otros componentes del ecosistema BC como, por ejemplo, los oráculos.

De forma general, la configuración realizada para la validación del esquema permite una simulación realista del entorno de producción, posibilita pruebas exhaustivas de interoperabilidad entre los diferentes componentes del sistema. La combinación de recursos locales y en la nube optimiza la eficiencia y escalabilidad de nuestro entorno de pruebas, y suministra una plataforma robusta para la validación de la dApp en condiciones cercanas a las de producción.

4.4.1 Prueba de tiempo y costo de ejecución de funciones de SC

La configuración establecida para el esquema propuesto será sometida a una serie de simulaciones que tienen como objetivo principal medir dos parámetros críticos de evaluación en las interacciones directas por parte de cliente y proveedor en el esquema: los costos por transacción asociados y los tiempos de ejecución. En estas simulaciones se ejecutan funciones de forma exitosa siguiendo flujos de operación lógicos que se plantean en el esquema propuesto a partir del uso de la librería Web3 de Python como SDK.

La Figura 34 presenta un diagrama detallado que ilustra los procedimientos a seguir durante estas simulaciones. Este diagrama es la guía visual para comprender el flujo para la obtención de los parámetros de evaluación desde la PC local. El estudio se centra en analizar los parámetros mencionados (costos y tiempos) para las funciones específicas los SC que los actores del esquema pueden ejecutar directamente a través de la aplicación descentralizada. Estas funciones se implementaron durante las pruebas locales y son particularmente relevantes ya que representan las interacciones clave de los usuarios con el sistema propuesto y su ejecución impacta directamente en la experiencia de los usuarios de la dApp. A continuación, se describen algunas funciones:

- constructor() de SC Mercado: Con la realización de esta función, ejecutada por un proveedor, se efectúa el despliegue del SC Mercado. Esta es la primera función para desplegar ya que inicializa la operación del esquema.

- AddProvider() de SC Mercado: Al ejecutarla se añade un nuevo proveedor a la lista de proveedores del SC Mercado, solamente puede hacerlo otro proveedor.

- createCustomSLA() de SC Mercado: Realiza el despliegue de un nuevo SC SLA inactivo, y posteriormente efectúa el despliegue del SC Subasta asociado a este SLA. Solamente un proveedor previamente registrado puede llamar a esta función.

- bid() de SC Subasta: Esta función puede ser llamada tanto por un cliente como por un proveedor para hacer una oferta por un SLA en específico.

- auctionEnd() de SC Subasta: Al llamar a esta función, si ha culminado el tiempo de subasta y no hay ofertas superiores a la oferta mínima se termina la misma sin cliente asociado al SC SLA. Si culminó la subasta y hubo ofertas por encima de la mínima, el valor de la máxima oferta es transferido al SC SLA correspondiente y se le asigna el cliente que realizó esa oferta. Según el esquema esta función se ejecuta de manera automática desde la dApp, también puede ser llamada por el proveedor.

Como primer paso en el esquema de la Figura 34, se tiene la definición de la función a ejecutar para determinar su tiempo y costo de ejecución. El procedimiento difiere según la naturaleza de la función, ya sea un constructor o una función regular del contrato. Para poder ejecutar las funciones se implementaron flujos de ejecución que simulan escenarios en que puedan ser llamadas siguiendo la lógica del esquema propuesto, pues en caso contrario podría no ser posible ejecutarlas.

Teniendo en cuenta que una función “constructor” despliega un nuevo contrato y el contrato no existe en la BC, para su ejecución se utiliza el contrato compilado en conjunto con su Interfaz de Aplicación Binaria (ABI, por las siglas del término en inglés, *Application Binary Interface*). El contrato compilado es un conjunto de bytes que contiene las instrucciones del SC a desplegar, y que la EVM puede comprender. El ABI define la interfaz del contrato, sus funciones, su interacción con ellas, asimismo, define los parámetros que necesita para inicializar el contrato. Para la ejecución y análisis del resto de las funciones, que no son de tipo constructor, se requiere la dirección del contracto desplegado, que identifica de manera única la instancia específica del contrato en la BC en conjunto con la ABI.

Habrá que recordar que la ejecución de una función es un tipo de transacción en el entorno de los SC y BC. Antes de realizar una transacción de este tipo, en el esquema de evaluación (ver Figura 34) se procede con la captura del tiempo actual. Luego se realiza la transacción con los parámetros requeridos por la función. La BC responderá por medio del proveedor de nodos con un recibo el cual incluye el estado de la

transacción en la BC. De ser ‘1’ el estado del recibo, la transacción se asume como exitosa. En caso opuesto esto significa la ocurrencia de un error en la transacción.

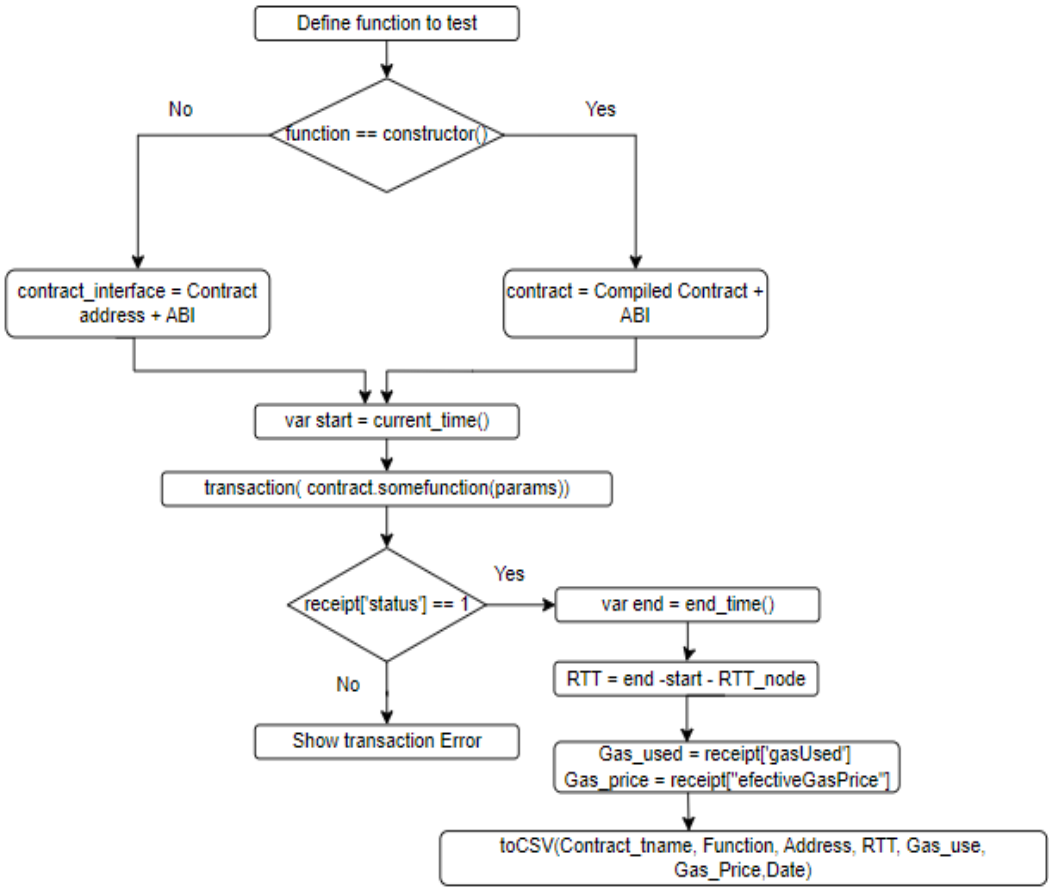

Figura 34. Diagrama de obtención de tiempo y costo en ejecución de funciones de SC [Elaboración Propia].

Uno de los intereses principales radica en evaluar el desempeño de la BC en relación con el tiempo invertido para ejecutar sus funciones. Al obtenerse una ejecución exitosa de la función a probar, se captura el tiempo de terminación y se le resta al tiempo de inicio. Como este tiempo incluye el tiempo que demora la solicitud de transacción al proveedor de nodos se resta también el tiempo de ida y vuelta de la solicitud entre proveedor de nodos y PC local. Del recibo de una transacción exitosa también se obtienen dos parámetros que permiten calcular el valor de la tarifa de transacción que representa el costo correspondiente a la ejecución de la función. Estos parámetros son el *gas* usado y el precio efectivo del *gas*. Su producto define el valor de la tarifa de transacción.

Luego de obtener el tiempo de ejecución del contrato se almacenan los datos relativos a cada transacción en un archivo de valores separados por comas (CSV, por las siglas del término en inglés, *Comma-Separated Values*). Los datos almacenados son el tipo de contrato que puede ser SC Mercado o SC Subasta, el nombre

de la función, la dirección del contrato de la función llamada o la dirección del contrato desplegado para el caso de la función “constructor”, el valor del tiempo de ejecución de la función y el precio de la tarifa de transacción.

Para la implementación del diagrama de obtención de los parámetros, se utilizó la biblioteca Web3 de *Python*, también es posible usar scripts en *Solidity* de forma similar que se utilizó para las pruebas locales. Esta decisión se justifica con el hecho de que en *Solidity* la medición de los valores de tiempo de ejecución no son fácilmente accesibles. *Solidity* tiene acceso limitado a información sobre el tiempo de ejecución de la transacción y no tiene funciones nativas para medir el tiempo con precisión a diferencia de *Python*.

El experimento realizado abarcó la ejecución iterativa de cinco funciones objetivo. Cada función se somete a veinte iteraciones en dos entornos de prueba distintos: la *testnet Polygon Mumbai* y *Ethereum Sepolia*. Se implementó el procedimiento previamente descrito para la captura sistemática de datos transaccionales, los resultados correspondientes a cada *testnet* específica se almacenaron en dos conjuntos de datos diferenciados y se almacenaron en formato CSV.

4.4.2 Resultados de pruebas de ejecución de funciones de SC

Las cinco funciones clave de interacción entre los actores del esquema descritas en la sección 4.4.1 se ejecutaron en dos redes BC de prueba (*testnets*) diferentes. La Figura 35 y la Figura 36 presentan diagramas de caja o *boxplot* que ilustran la distribución de las tarifas de transacción para las funciones, ya descritas. La Figura 35 muestra los resultados obtenidos en la *testnet Ethereum Sepolia*, mientras que la Figura 35 presenta los resultados obtenidos en la *testnet Polygon Mumbai*.

De forma complementaria, las Figuras 37 y 38 muestran diagramas de caja o *boxplot* que representan los tiempos de ejecución de estas mismas funciones. La Figura 37 corresponde a los tiempos de ejecución en la *testnet Ethereum Sepolia*, y la Figura 38 muestra los tiempos de ejecución en la *testnet Polygon Mumbai*.

Un *boxplot* es una representación gráfica de la distribución de un conjunto de datos numéricos. Muestra la mediana, los cuartiles y los valores extremos de los datos en una sola imagen. La "caja" central representa el 50% de los datos centrales, con una línea que indica la mediana. Los "bigotes" se extienden para mostrar el rango de los datos, se excluyen valores atípicos, que se representan como puntos

individuales. Este tipo de gráfico es útil para visualizar rápidamente la dispersión y simetría de los datos, así como para identificar valores inusuales.

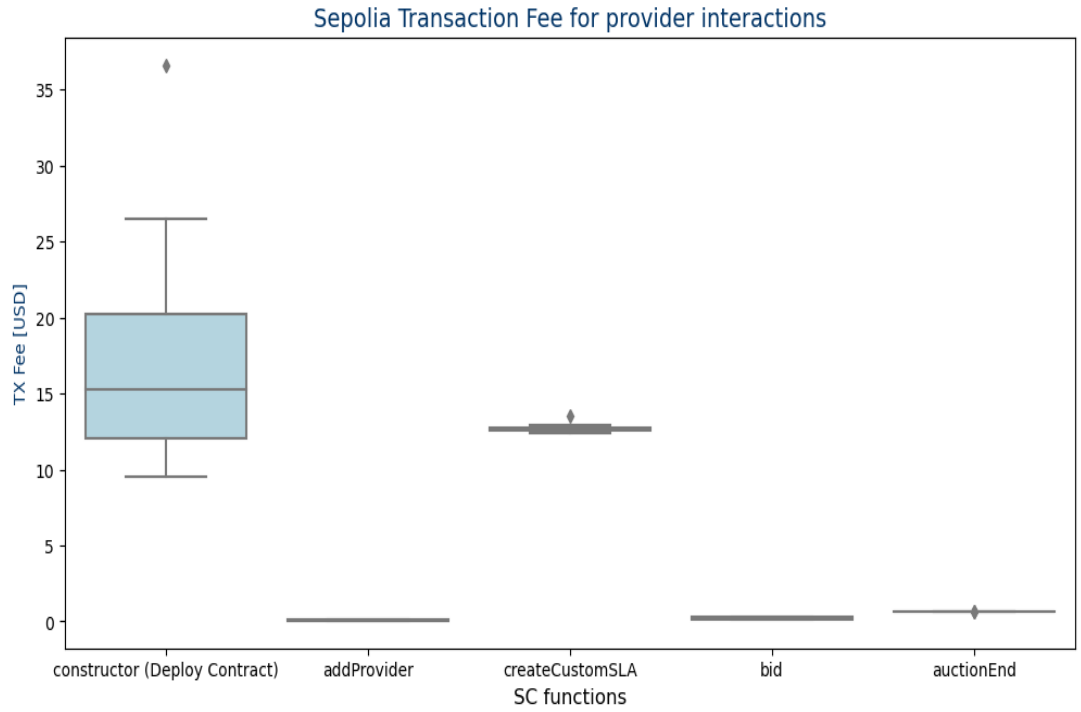

Figura 35. Diagrama de caja de tarifa de transacción en USD para 5 funciones de SC en la *testnet Sepolia* [Elaboración Propia].

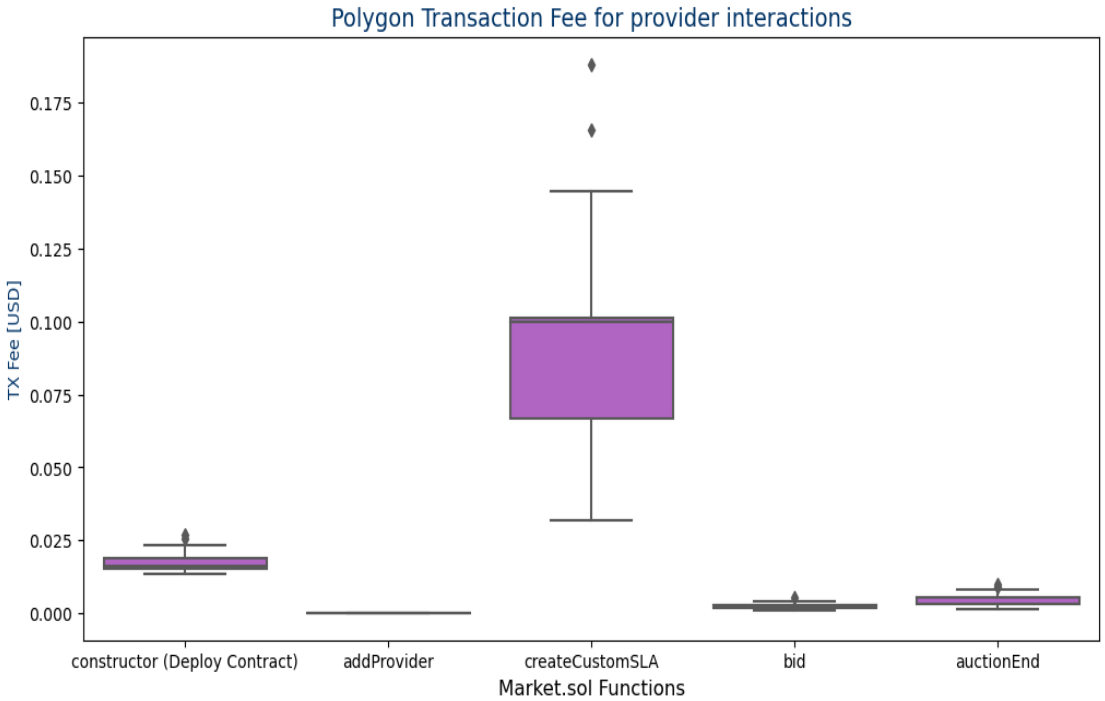

Figura 36. Diagrama de caja para tarifas de transacción en USD para 5 funciones de SC en la *testnet Mumbai* [Elaboración Propia].

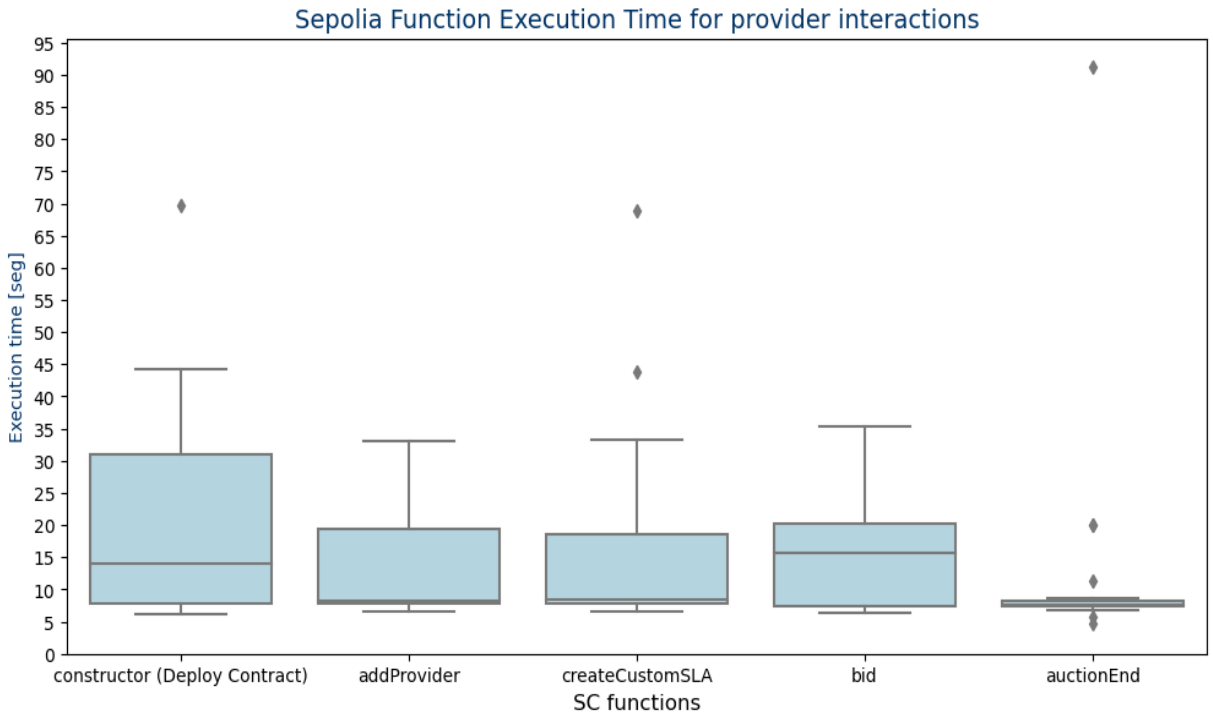

Figura 37. Diagrama de caja para tiempo de ejecución de 5 funciones de SC en la *testnet Sepolia* [Elaboración Propia].

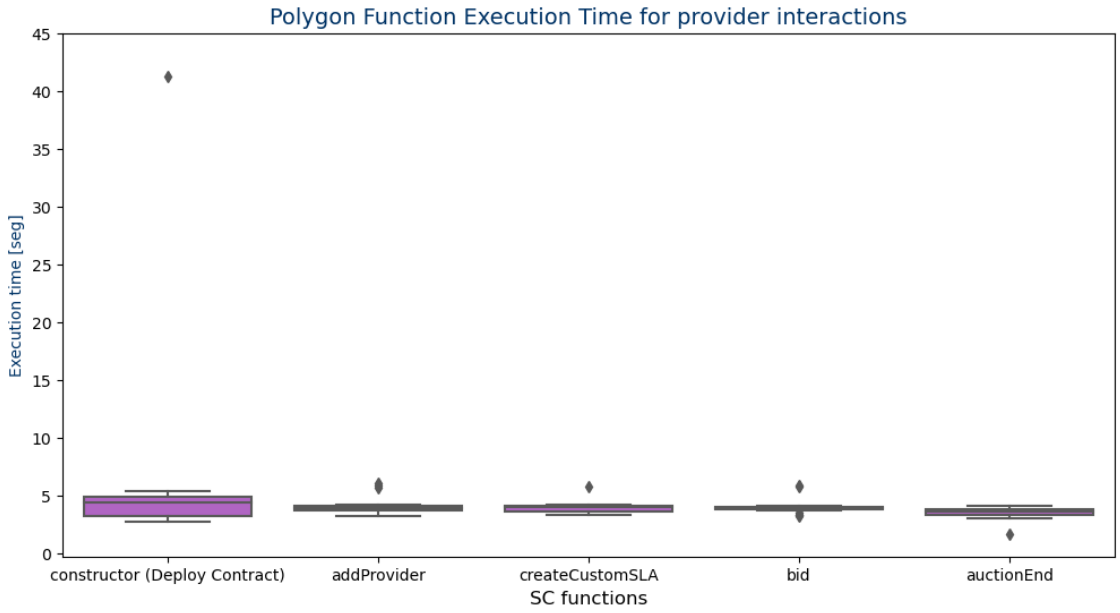

Figura 38. Diagrama de caja para tiempo de ejecución de 5 funciones de SC en la *testnet Mumbai* [Elaboración Propia].

Estos gráficos permiten una comparación visual directa del tiempo de ejecución y los costos asociados a cada función en ambas redes de prueba, lo cual facilita el análisis de la eficiencia y viabilidad de las

operaciones en cada BC. La discusión de estos resultados se realiza en la sección 4.5.

4.4.3 Prueba de tiempo de respuesta a solicitud de información externa

En el esquema propuesto se pretende monitorizar los SLA de NS y para ello se deben extraer métricas del mundo exterior, en la Figura 39 se muestra la interacción de las partes involucradas en ese proceso. La determinación del período mínimo de extracción de estas métricas tiene una importancia crítica, pues establece los límites temporales viables para el periodo de monitoreo de creación del SLA. Esta consideración previene la formulación de contratos con períodos de medición excesivamente breves, lo cual podría comprometer la capacidad del oráculo para proporcionar las métricas solicitadas de manera precisa. La definición de este umbral temporal no solo optimiza la eficacia del sistema de monitorización, sino que también garantiza la integridad y factibilidad de los acuerdos establecidos en la plataforma BC.

Se diseñó un experimento para determinar el período mínimo de solicitud de datos en cada *testnet*. Este proceso implicó la creación y despliegue de un SC especializado llamado APIConsumer, cuya función principal es interactuar con el oráculo distribuido *Chainlink* para solicitar datos de una API simulada del proveedor.

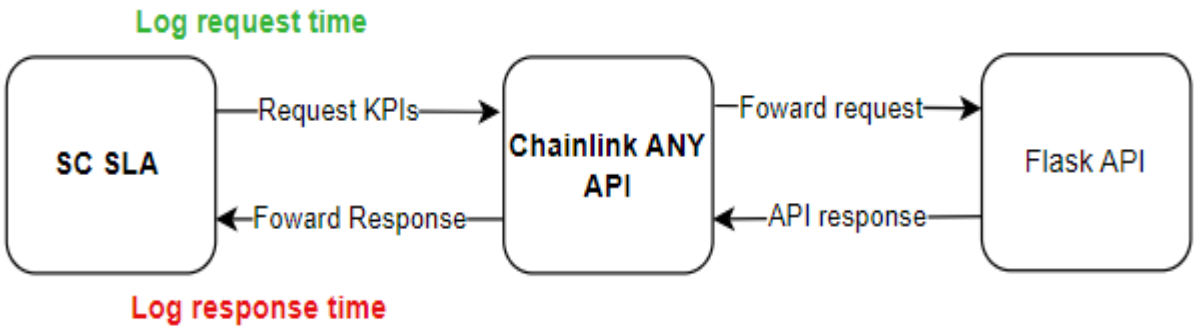

Figura 39. Extracción de KPI de la API del proveedor [Elaboración Propia].

El SC se configuró con parámetros específicos para cada *testnet*, incluyendo la dirección digital del oráculo correspondiente, la especificación del tipo de trabajo que llevaría a cabo el oráculo (que sería hacer una solicitud HTTP) y la URL de acceso a las métricas de monitorización vía API. La arquitectura del contrato incluye dos funciones clave:

- Una función de solicitud: Al ser invocada, emite un evento *requestVolumeData*, registrando en los logs de la BC el tiempo exacto de la solicitud en formato Unix (tiempo de solicitud).

- Una función de respuesta: Es activada por el oráculo configurado, emite un evento *fullfillOracleRequest2*, captura en los logs del SC varios parámetros e incluye el tiempo Unix de recepción de la respuesta (tiempo de respuesta).

La diferencia temporal entre estos dos eventos proporciona una medida precisa del intervalo entre la solicitud y la recepción de datos como se muestra en la Figura 39. Este intervalo se utiliza como base para establecer el período mínimo viable para la solicitud de métricas de monitorización.

Se utilizaron exploradores de bloques (EB) para conocer los dos valores de tiempo correspondientes a los eventos descritos anteriormente. Los EB son herramientas en línea que permiten a los usuarios visualizar y analizar la información contenida en una BC (Kuzuno & Karam, 2017). Para el caso de *Sepolia* se utiliza el EB *etherscan* y para el caso de *Mumbai* el EB *polygonscan*.

Con estas herramientas se puede crear filtros para obtener la información relacionada con un contrato en específico previamente desplegado. En estos casos interesa filtrar los eventos relacionados con el SC APIConsumer y conocer los valores de los tiempos de solicitud y respuesta. Con la diferencia de estos valores se obtiene el tiempo mínimo posible para un período de medición de KPI en un SLA.

4.4.4 Resultados para prueba de solicitud de información externa

La prueba se realizó llamando a la función de solicitud 20 veces a intervalos espaciados de 3 min desde la PC local en ambas *testnet*. El contrato APIConsumer se desplegó y como efecto se registraron eventos vinculados al SC APIConsumer en los logs de la BC. En la Figura 40 se puede apreciar una muestra del resultado de filtrar los eventos del contrato desplegado en *Ethereum Sepolia* usando *Etherescan*.

De forma análoga en la Figura 41 se obtiene la información de los eventos asociados al contrato desplegado en *Polygon Mumbai* usando *Polygonscan*. La diferencia entre el tiempo en que se recibió la respuesta y el tiempo en que se realizó la solicitud resultó para *Ethereum Sepolia* ser de 12 segundos en las 20 ocasiones. Para el caso de *Polygon Mumbai* fue de 6 segundos.

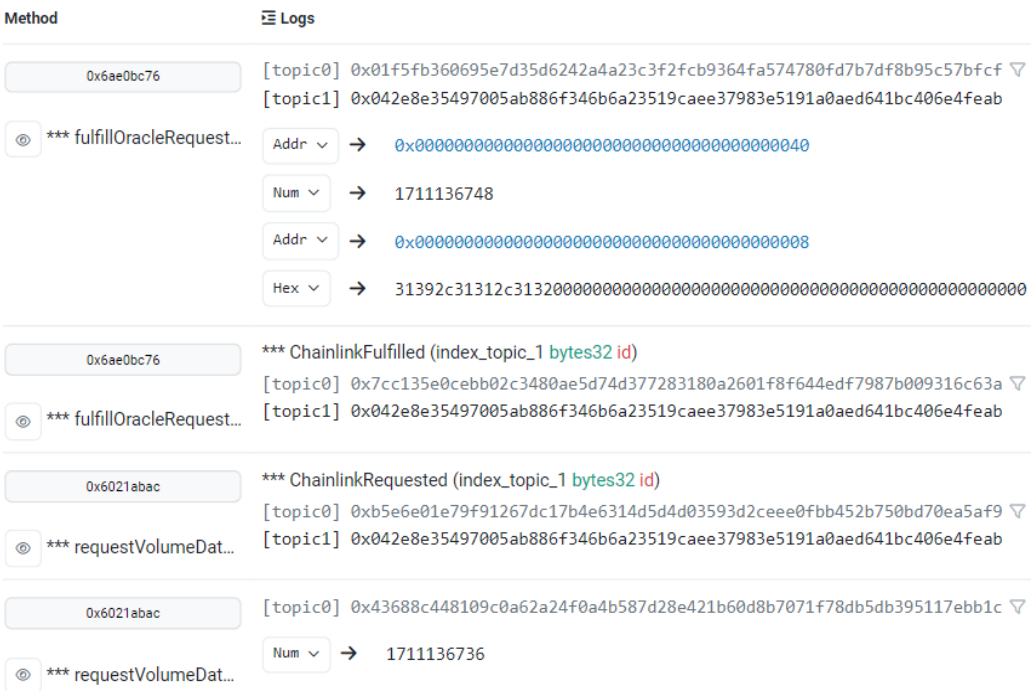

Figura 40. Log de eventos en interacción SC-Oráculo en *Ethereum Sepolia* [Elaboración propia].

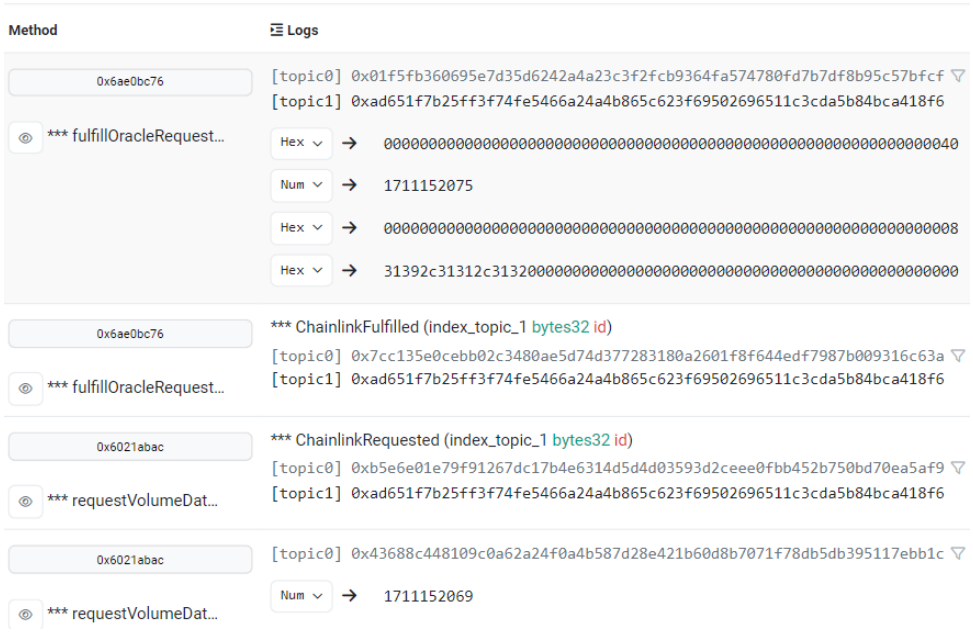

Figura 41. Log de eventos en interacción SC-Oráculo en *Polygon Mumbai* [Elaboración propia].

4.5 Discusión de resultados

En las secciones 4.3 y 4.4 se presentó tanto la metodología como las pruebas realizadas en el ámbito local y en *testnet*. En esta sección se analizan y discuten los resultados obtenidos en cuanto a cobertura de

pruebas y ejecución exitosa de las mismas, se discuten y analizan también los resultados sobre el tiempo y costos de ejecución de funciones de SC y la prueba llevada a cabo para determinar el periodo mínimo viable para la obtención de KPI para ambos escenarios.

4.5.1 Discusión de resultados de pruebas locales

Con respecto a las pruebas locales descritas en la sección 4.3 se construyeron 32 test los cuales resultaron exitosos y se demuestra que el flujo de operación de los SC se implementó correctamente. Como se observa en la Figura 31, los SC pasaron todos los test en un tiempo de ejecución de prueba de 1 segundo.

Referente a la cobertura de pruebas para los SC del esquema en la mayoría de los casos se sobrepasa el 70%. Hay espacio para mejorar la cobertura en SC Mercado y algunas áreas específicas de los otros contratos. A partir de estos resultados reflejados en la Figura 30, se puede interpretar que indican un buen nivel de pruebas, que las partes críticas del código están siendo probadas adecuadamente especialmente para SC Subasta y SC SLA.

No se probaron las funciones que se encargan solamente de devolver una variable de estado, ya que en muchos casos no intervienen en el flujo del esquema propuesto. Es Market.sol quien mayor cantidad de este tipo de funciones contiene y esa la razón por la que no se acerca al 100% la cobertura de funciones como si lo hacen el resto de los porcientos.

La implementación meticulosa de pruebas unitarias y de integración en un entorno de desarrollo local, previo al despliegue en una red de pruebas (*testnet*) o en una BC en producción, se revela como una práctica fundamental e ineludible en el desarrollo de contratos inteligentes. Su implementación no solo mejora sustancialmente la calidad, confiabilidad y seguridad de los contratos, sino que también minimiza de manera significativa los riesgos y costos asociados con la detección tardía de errores.

El despliegue de un contrato en una red BC real implica gastos considerables en forma de tarifas de transacción. Debido a la naturaleza inmutable de la BC, una vez desplegado el contrato no puede ser modificado directamente. Esta característica enfatiza la importancia de garantizar la corrección del código antes del despliegue. Las pruebas locales permiten identificar y corregir problemas de manera automática y temprana en el ciclo de desarrollo. Esto no solo ahorra tiempo, sino que también previene potenciales pérdidas financieras y reputacionales asociadas con errores en contratos desplegados. La combinación de

pruebas unitarias y de integración asegura una cobertura amplia del código, abarcando desde la funcionalidad de componentes individuales hasta las interacciones complejas entre múltiples partes del sistema.

Un conjunto robusto de pruebas facilita la refactorización segura del código y simplifica el mantenimiento a largo plazo y permite realizar cambios con confianza. Además, pruebas bien diseñadas actúan como una forma de documentación viva e ilustrando el comportamiento esperado del contrato en diversos escenarios.

La inversión de tiempo y recursos en el desarrollo de pruebas exhaustivas no solo constituye una buena práctica de ingeniería de software, sino que se erige como un imperativo en el ecosistema de contratos inteligentes. Esta práctica no solo optimiza el proceso de desarrollo, sino que también salvaguarda los activos digitales y la integridad de las operaciones basadas en BC.

Con esta prueba local se verifica el cumplimiento de SLA en un entorno local simulado, y se cumple de esta manera con uno de los objetivos del presente trabajo: que el esquema verifique el cumplimiento de SLA.

4.5.2 Discusión de resultados de ejecución de funciones en testnet

En las gráficas de las Figuras 35 a la 38 se observa que existen variaciones con respecto a los tiempos y costos de ejecución de las funciones. Estas variaciones están en dependencia de la estabilidad de la red BC específica, del estado de congestión que presente, el precio del *gas* por la demanda en la red o el tipo de transacción a realizar. De forma general se puede apreciar mayor estabilidad para el caso de la red *Polygon Mumbai* ya que presenta una variación más pequeña para todos los casos.

Los costos se presentan en dólares (USD), sin embargo, los valores de las transacciones fueron obtenidos en unidades de las *faucet coins* propias de cada *testnet*: *Matic* en el caso de *Polygon Mumbai* y *Ethereum* en el caso de *Sepolia*. Se hizo la conversión a USD en dependencia de la tasa de cambio al momento de las transacciones respectivamente. Se puede observar que los costos de transacción son mucho menores para la red *Polygon Mumbai* y presentan menor dispersión. De igual manera, para la red *Mumbai* el tiempo de ejecución en la mayoría de los casos es cercano a su media y menor que los tiempos de ejecución de las funciones en la red *Sepolia*.

Como es lógico se obtuvo que *Polygon Mumbai* ofrece mayor estabilidad en tiempos de ejecución y costos de transacción comparado con *Ethereum Sepolia* debido a su diseño técnico específico. Como solución de capa 2, *Polygon Mumbai* opera como una cadena lateral de *Ethereum*, utiliza un mecanismo de consenso PoA en combinación con PoS el cual es más eficiente. Esto permite a *Polygon Mumbai* generar bloques en la BC con un periodo menor, a diferencia de *Ethereum Sepolia*, acelerando significativamente las transacciones.

Las optimizaciones de *gas* implementadas en *Polygon Mumbai* resultan en tarifas de transacción más bajas y estables. En contraste, *Sepolia*, al emular más fielmente la red principal de *Ethereum*, puede experimentar mayores fluctuaciones en los costos de *gas* y tiempos de ejecución, especialmente durante periodos de alta actividad.

Esta combinación de factores técnicos hace que *Mumbai* ofrezca un entorno más estable y económico para la ejecución de contratos inteligentes y transacciones. Con los resultados obtenidos se muestra la importancia de la elección de una BC para el rendimiento de un sistema que la incluya y se logra uno de los objetivos específicos de esta tesis que es analizar el desempeño del esquema propuesto en diferentes BC. Con este experimento se cubre un aspecto que no se trata en otros trabajos como se mencionó en la sección 1.2, donde no se comparan las soluciones de gestión de SLA usando SC para diferentes BC.

La efectividad en las llamadas a estas funciones mediante la programación de casos o flujos para su ejecución mediante el uso de SDK demuestra la capacidad del esquema para realizar el establecimiento, monitorización y gestión de pagos. De esta forma también se valida la implementación del esquema propuesto en una *testnet*.

4.5.3 Discusión de resultados de prueba de solicitud de información externa

El experimento para determinar el periodo mínimo viable según la BC utilizada arrojó resultados significativos: 12 segundos para *Ethereum Sepolia* y 6 segundos para *Polygon Mumbai*. Estos hallazgos demuestran que los periodos de monitorización de SLA comúnmente empleados en el entorno de NS, como el intervalo de 1 minuto, son perfectamente aplicables al esquema propuesto. Es importante destacar que estos periodos inferiores a 1 minuto son para la extracción de métricas de monitorización, lo cual significa, que con esta prueba también se evalúa la factibilidad de la medición de varias métricas de forma simultánea. La notable diferencia en los tiempos mínimos entre las dos redes sugiere una mayor

eficiencia en *Polygon Mumbai*, lo que podría ser un factor decisivo en la elección de la plataforma según los requisitos específicos del proyecto.

Estos resultados validan la viabilidad del oráculo para la extracción de parámetros externos, de manera que se garantizan tiempos de respuesta adecuados para una monitorización efectiva. La capacidad de operar dentro de estos márgenes temporales no solo cumple con los estándares actuales de monitorización de SLA, sino que también ofrece la flexibilidad necesaria para adaptarse a diversos escenarios de uso.

4.6 Conclusiones

La validación del esquema propuesto de gestión de SLA de NS se basa en la dupla BC y SC. A lo largo del capítulo se realizaron pruebas que muestran la viabilidad y efectividad en el flujo de operación del esquema propuesto para integrar etapas propias del ciclo de vida de un SLA de forma automatizada en una BC local y en las *testnet Ethereum Sepolia* y *Polygon Mumbai*. Estas pruebas proporcionaron resultados que fueron analizados de forma exhaustiva con el fin de alcanzar un mayor entendimiento relativo al desempeño del esquema según la BC utilizada. Por tanto, se puede concluir que:

- ✓ Un ecosistema de herramientas constituido por un editor de código con funcionalidades extensibles unido a *Git* para el control de versiones y un *framework* de trabajo como *Foundry*, acelera y aumenta la eficiencia en el desarrollo, implementación y realización de pruebas sobre SC.

- ✓ El desarrollo de pruebas unitarias y de integración de forma local en paralelo con la codificación de los SC, permite la detección temprana de errores en el flujo de operación simulado del esquema propuesto. Al tener un porcentaje adecuado de cobertura de pruebas y varios *test* exitosos que simulen el flujo de ejecución del esquema, se mejora significativamente la calidad, confiabilidad y seguridad de los SC.

- ✓ Los SC implementados contienen funciones que fueron ejecutadas exitosamente utilizando flujos de ejecución en *testnet*. El resultado de ejecución sin errores para estos flujos valida la capacidad del esquema para establecer acuerdos, monitorearlos y realizar pagos de forma automatizada.

- ✓ La selección de una BC en específico impacta directamente en el desempeño de un esquema que utilice BC lo que se puede apreciar en los resultados de ejecución de pruebas de tiempo y costo de ejecución de funciones de SC donde se obtuvo mejor desempeño para la *testnet Polygon Mumbai*.

- ✓ Para las pruebas en *testnet* se utilizaron varias métricas de monitoreo para ser extraídas desde el exterior a la BC, sin embargo, esto no tuvo un impacto negativo en la selección de un periodo mínimo de medición de 1 minuto pues la latencia introducida es de apenas 6 y 12 seg para *Polygon Mumbai* y *Ethereum Sepolia* respectivamente.

- ✓ El uso de oráculo *Chainlink* es viable para un periodo de medición estándar de 1 minuto.

Capítulo 5. Conclusiones y recomendaciones

En este trabajo se reconoce que existen desafíos significativos en la gestión de los SLA de NS los cuales pudieran resolverse con la utilización de contratos inteligentes (SC) y Blockchain (BC). Por esta razón se realizó una revisión de la literatura actualizada para sentar las bases teóricas y conocer los antecedentes relativos al uso de SC y BC en el entorno de gestión de Acuerdos de Nivel de Servicio (SLA) de *Network Slices*. Se realizó un estudio y análisis relativos a la arquitectura de la red 5G, la tecnología de segmentación de red NS, el ciclo de vida del SLA y las tecnologías de la Web3 entre las cuales se encuentran BC y SC. Se planteó que NS permite una arquitectura basada en servicios para la red 5G y se prevé será parte de la arquitectura en redes 6G. Una arquitectura SBA más eficiente con respecto a la arquitectura monolítica de 4G. En el estudio realizado se destaca el papel de NS en redes 5G/B5B para el surgimiento de un nuevo mercado con múltiples interesados en arrendar segmentos de red mediante el establecimiento de SLA, lo que conlleva nuevos retos para la gestión de SLA.

Como resultado de la revisión bibliográfica y análisis tecnológico llevado a cabo se diseñó y propuso un esquema para la gestión de SLA de NS que aprovecha las características de inmutabilidad de BC y la posibilidad de desplegar SC como piezas de código que automatizan el ciclo de vida de SLA sin la participación de intermediarios. Con el esquema se aumenta la transparencia desde la perspectiva del cliente, se reducen los costos operativos del proveedor e incrementa la automatización en la gestión de SLA de NS.

En la validación del esquema propuesto se verificó el correcto flujo de operación del esquema en la ejecución del ciclo de vida de los SLA mediante pruebas en una BC local. El resultado positivo de estas pruebas demuestra la capacidad del esquema para comprobar la ejecución de SLA.

Como parte de la validación también se analizó el impacto de la selección de determinada BC de prueba en el desempeño del esquema y se obtuvo como resultado que *Polygon Mumbai* es más eficiente que *Ethereum Sepolia* en lo que respecta a costos y tiempo de ejecución de funciones de SC. Se afirma que llevar a cabo la ejecución exitosa y sin errores de estas funciones al implementar los flujos de operación del esquema de forma simulada demuestra una implementación robusta del mismo.

Con respecto al sistema propuesto se puede concluir que cubre aspectos que no son abordados en la literatura en los trabajos anteriores analizados pues se incluye *Chainlink* como oráculo distribuido para aumentar la confiabilidad y disponibilidad de las métricas de monitoreo de SLA, y se integran

funcionalidades que los SC tienen capacidad de llevar a cabo como la intermediación en la negociación, la monitorización y la gestión de pagos y recompensa. La carencia de análisis comparativos del impacto según el tipo de BC y el tipo de mecanismo de consenso utilizado en trabajos previos es superada en este trabajo ya que se realizaron experimentos para determinar las diferencias de resultados de ejecución de funciones en diferentes BC.

5.1 Contribuciones al conocimiento

El desarrollo de esta investigación contribuye a mejorar la gestión de SLA para el arrendamiento de NS al proponerse un esquema basado en SC y BC. A partir del análisis de las tecnologías involucradas, la concepción del esquema y su validación se tiene que los aportes de este trabajo son:

Se identifican problemas relacionados con la gestión tradicional de los SLA en NS.

Se identifican limitaciones de trabajos previos que incluyen SC y BC en la gestión de SLA.

Se detallan las diferencias de implementación de un sistema de gestión que utilizan BC públicas frente a los que incorporan BC privadas.

Se propone un esquema que automatiza el ciclo de vida de SLA para el caso de arrendamiento de NS en entornos 5G/B5G de forma segura, transparente y distribuida.

Se aportan criterios de selección de tecnologías para el sistema propuesto.

Se desarrolla una metodología para el desarrollo de SC del esquema de gestión de SLA concebido.

Se desarrollan dos entornos de simulación uno local y otro en *testnets* para la validación del sistema propuesto.

5.2 Limitaciones de la investigación

Como todo trabajo de investigación, existen situaciones que se salen del alcance de la misma. A continuación, se describen algunas de las limitantes de la investigación:

- ✓ A pesar de que el sistema concebido posibilita la intermediación en el establecimiento de SLA, el monitoreo de KPI para NS y la realización de pagos de forma automatizada no tiene en consideración la asignación directa de recursos en una red 5G o la creación de un NS real, dado que no se trabaja con una API real de proveedor de NS sino con una abstracción de ella.

- ✓ En este trabajo la validación se realiza de forma local y en *testnet*, más no en una BC real lo que constituiría una mejor demostración de la capacidad de la arquitectura propuesta para la gestión de SLA de NS, esto conllevaría un gasto recursos excesivo, innecesarios para la investigación.

- ✓ Por razones presupuestales se trabajó en la capa gratuita del proveedor de nodos *Alchemy,* la cual no soporta un número grande de peticiones, que es requerido para evaluar la cantidad de transacciones que puede soportar la BC para el esquema propuesto, dado que es un indicador notable de su desempeño

- ✓ El estudio comparativo del desempeño de la arquitectura propuesta no incluye el análisis del rendimiento del esquema en una BC privada, dada la complejidad y tiempo requeridos para una implementación de este tipo, se propone como tarea futura.

5.3 Trabajo futuro

Después del estudio realizado y el diseño y validación del esquema propuesto se pueden considerar los siguientes trabajos futuros:

- ✓ Realizar la simulación de una red 5G/B5G y añadir la lógica para la asignación de recursos en el esquema propuesto. De esta forma se podría analizar el comportamiento de la arquitectura propuesta para la gestión de SLA en una red 5G/B5G simulada y no sobre una abstracción de ella, con la funcionalidad adicional de asignación de recursos por parte de los SC.

- ✓ Implementar nodos propios de la BC a utilizar, es decir, de evitar la dependencia de proveedores de nodos. Con ello se podría analizar el volumen de transacciones posibles y establecer su límite.

- ✓ Construir una interfaz gráfica para dApp en vistas de facilitar su uso y posibilitar la actividad interactiva a futuros usuarios.

- ✓ Ampliar el estudio comparativo del desempeño del esquema según la BC utilizada en cuanto a la utilización de diferentes BC, reales y privadas.

Literatura citada

3GPP. (2019). 3GGP 23501 System Architecture for the 5G System. Vol. 16.0.2(Número Release 16, p. 308). https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.06.00_60/ts_123501v160600p.pdf

5gAmericas. (2016). Nertwork Slicing for 5G Networks and services. https://www.5gamericas.org/wp-content/uploads/2019/07/5G_Americas_Network_Slicing_11.21_Final.pdf

Afraz, N., Wilhelmi, F., Ahmadi, H., & Ruffini, M. (2023). Blockchain and Smart Contracts for Telecommunications: Requirements vs. Cost Analysis. *IEEE* Access, *11*, 95653–95666. https://doi.org/10.1109/ACCESS.2023.3309423

Alabdulwahhab, F. A. (2018). Web 3.0: The Decentralized Web Blockchain networks and Protocol Innovation. 1st International Conference on Computer Applications and Information Security, ICCAIS 2018, 1–4. https://doi.org/10.1109/CAIS.2018.8441990

Amazon Web Services, Inc. (2024). ¿Qué es una API de RESTful? https://aws.amazon.com/es/what-is/restful-api/

Azzahra, Z. F., & Nugraha, I. G. B. B. (2023). Service-Level Agreement Management with Blockchain-based Smart Contract to Improve the Quality of IT Service Management. 260–266. https://doi.org/10.1145/3587828.3587867

Baniaș, O., Florea, D., Gyalai, R., & Curiac, D. I. (2021). Automated specification-based testing of REST APIs. *Sensors*, *21*(16). https://doi.org/10.3390/s21165375

Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System. CoRR. http://arxiv.org/abs/1407.3561

Beniiche, A. (2020). A Study of Blockchain Oracles. http://arxiv.org/abs/2004.07140

Buterin, V. (2014). Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

Chainlink, Inc. (2024a). Chainlink Automation. https://docs.chain.link/chainlink-automation

Chainlink, Inc. (2024b, January 12). What Is an Oracle in Blockchain? » Explained | Chainlink. https://chain.link/education/blockchain-oracles#toc-first

Chang, P. (2015). Fundamentals of HTTP Fundamentals of HTTP White Paper. https://www.f5.com/content/dam/f5/corp/global/pdf/white-papers/http-fundamentals-wp.pdf

Charismiadis, A. S., Salcines, J. M., Tsolkas, D., Guillen, D. A., & Rodrigo, J. G. (2023). The 3GPP Common API framework: Open-source release and application use cases. 2023 Joint European

Conference on Networks and Communications and 6G Summit, EuCNC/6G Summit 2023, 472–477. https://doi.org/10.1109/EuCNC/6GSummit58263.2023.10188337

Costa-Pérez, X., Sciancalepore, V., Zanzi, L., & Albanese, A. (2023). Blockchain for Mobile Networks. In Blockchains (pp. 185–213). Wiley. https://doi.org/10.1002/9781119781042.ch7

Damjan, M. (2018). The interface between blockchain and the real world. Ragion Pratica, 2018(2), 379–406. https://doi.org/10.1415/91545

De Brito Gonçalves, J. P., Lima Gomes, R., Da Silva Villaca, R., Municio, E., & Marquez-Barja, J. (2020). A Quality of Service Compliance System Empowered by Smart Contracts and Oracles. Proceedings - 2020 IEEE International Conference on Blockchain, Blockchain 2020, 532–538. https://doi.org/10.1109/Blockchain50366.2020.00077

Dense Networks. (2024). 4g/5g. https://densenetworks.com/4g-5g

Dominguez, G., & Monzon Baeza, V. (2024). Desarrollo de una aplicación descentralizada basada en blockchain para el gobierno de una ciudad inteligente. http://hdl.handle.net/10609/149662

Fahim, S., Katibur Rahman, S., & Mahmood, S. (2023). Blockchain: A Comparative Study of Consensus Algorithms PoW, PoS, PoA, PoV. *International Journal of Mathematical Sciences and Computing*, *9*(3), 46–57. https://doi.org/10.5815/ijmsc.2023.03.04

Flask. (2024). Flask Documentation. https://flask.palletsprojects.com/en/2.3.x/

Foundry. (2024). Foundry Book . https://book.getfoundry.sh/

Gentilal, M., Martins, P., & Sousa, L. (2017). TrustZone-backed bitcoin wallet. ACM International Conference Proceeding Series, 25–28. https://doi.org/10.1145/3031836.3031841

Git. (2024). Git documentation. https://www.git-scm.com/doc

Gonzalez, M. A., De, A., Méndez, F., & Murillo, J. L. (2021). Implementación de un sistema para la atención temprana al desgaste psicológico en alumnos universitarios mediante técnicas de Blockchain e Inteligencia Artificial. https://oa.upm.es/67821/

Gonzalez-Franco, J., Lozano-Rizk, J., & Preciado, J. (2023). Reconocimiento de patrones en métricas de tráfico para el cumplimiento de los SLA (Service Level Agreement) en entorno de redes B5G (Beyond 5G) [Tesis de Maestría, Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California]. https://doi.org/10.13140/RG.2.2.19675.34083

GSM Association. (2017). An Introduction to Network Slicing. https://www.gsma.com/solutions-and-impact/technologies/networks/wp-content/uploads/2017/11/GSMA-An-Introduction-to-Network-Slicing.pdf

Guo, H., & Yu, X. (2022). A survey on blockchain technology and its security. *Blockchain: Research and Applications*, *3*(2). https://doi.org/10.1016/j.bcra.2022.100067

Habibi, M. A., Han, B., Nasimi, M., & Schotten, H. D. (2018). The Structure of Service Level Agreement of Slice-based 5G Network. http://arxiv.org/abs/1806.10426

Hu, J., & Liu, K. (2020). Raft consensus mechanism and the applications. *Journal of Physics: Conference Series*, *1544*(1). https://doi.org/10.1088/1742-6596/1544/1/012079

Huang, H., Lin, J., Zheng, B., Zheng, Z., & Bian, J. (2020). When Blockchain Meets Distributed File Systems: An Overview, Challenges, and Open Issues. *IEEE* Access, *8*, 50574–50586. https://doi.org/10.1109/ACCESS.2020.2979881

Jacobson, D., Brail, G., & Woods, D. (2011). APis: A Strategy Guide. O'Reilly Media, Inc.

Javed, F., & Mangues-Bafalluy, J. (2023). Blockchain-based SLA management for 6G networks. *Internet Technology Letters*, *7*(3), 472. https://doi.org/10.1002/itl2.472

Jin, L.-J., Machiraju, V., & Sahai, A. (2002). Analysis on Service Level Agreement of Web Services. https://www.csd.uoc.gr/~hy565/docs/pdfs/papers/HPL-2002-180.pdf

Kapassa, E., Touloupou, M., & Kyriazis, D. (2018). SLAs in 5G: A Complete Framework Facilitating VNF- and NS- Tailored SLAs Management. 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), 469–474. https://doi.org/10.1109/WAINA.2018.00130

Kapsoulis, N., Psychas, A., Litke, A., & Varvarigou, T. (2021). Reinforcing SLA consensus on blockchain. *Computers, 10*(12). https://doi.org/10.3390/computers10120159

Khan, S. N., Loukil, F., Ghedira-Guegan, C., Benkhelifa, E., & Bani-Hani, A. (2021). Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-Peer Networking and Applications, 14*(5). https://doi.org/10.1007/s12083-021-01127-0

Khanjari, S. Al, Arafeh, B., Day, K., & Alzeidi, N. (2013). Bandwidth borrowing-based QoS approach for adaptive call admission control in multiclass traffic wireless cellular networks. *International Journal of Communication Systems, 26*(7), 811–831. https://doi.org/10.1002/dac.1368

Kuzuno, H., & Karam, C. (2017). Blockchain explorer: An analytical process and investigation environment for bitcoin. ECrime Researchers Summit, ECrime, 9–16. https://doi.org/10.1109/ECRIME.2017.7945049

Leal, J. (2024). What is an RPC node? The Complete Guide (2024). https://blog.thirdweb.com/what-is-an-rpc-node/

Li, B., Deng, S., Yan, X., & Dustdar, S. (2022). The Confluence of Blockchain and 6G Network: Scenarios Analysis and Performance Assessment. http://arxiv.org/abs/2207.04744

Lin, S. Y., Zhang, L., Li, J., Ji, L. li, & Sun, Y. (2022). A survey of application research based on blockchain smart contract. *Wireless Networks, 28*(2), 635–690. https://doi.org/10.1007/s11276-021-02874-x

Massé, M. (2011). *REST API Design Rulebook*. O'Reilly Media, Inc.

Mayer, G. (2018). RESTful APIs for the 5G Service Based Architecture. *Journal of ICT Standardization, 6*(1), 101–116. https://doi.org/10.13052/jicts2245-800X.617

Meister, B. K., Henry, &, & Price, C. W. (2024). Gas Fees on the Ethereum Blockchain: From Foundations to Derivatives Valuations. https://arxiv.org/abs/2406.06524v1

Mogul, J. C. (2002). Clarifying the Fundamentals of HTTP. *Software: Practice and Experience, 34*(2), 103–134. https://doi.org/https://doi.org/10.1002/spe.573

Sangeeta, N., & Nam, S. Y. (2023). Blockchain and Interplanetary File System (IPFS)-Based Data Storage System for Vehicular Networks with Keyword Search Capability. *Electronics* (Switzerland), *12*(7). https://doi.org/10.3390/electronics12071545

Nadim Iqbal, F. (2023). A brief introduction to application programming interface (API). https://doi.org/10.5281/zenodo.10198423

Nguyen, T., Lovén, L., Partala, J., & Pirttikangas, S. (2021). The Intersection of Blockchain and 6G Technologies. In 6G Mobile Wireless Networks (pp. 393–417). Springer. https://doi.org/10.1007/978-3-030-72777-2_18

Nour, B., Ksentini, A., Herbaut, N., Frangoudis, P. A., & Moungla, H. (2019). A Blockchain-Based Network Slice Broker for 5G Services. *IEEE Networking Letters, 1*(3), 99–102. https://doi.org/10.1109/LNET.2019.2915117

Nzuva, S. (2019). Smart Contracts Implementation, Applications, Benefits, and Limitations. *Journal of Information Engineering and Applications, 9(*5), 63–75. https://doi.org/10.7176/JIEA

Papageorgious, A., Fernández-Fernández, A., & Ochoa-Aday, L. (2020). SLA Management Procedures in 5G Slicing-based Systems. European Conference on Networks and Communications (EuCNC), 7–11. https://doi.org/10.1109/EuCNC48522.2020.9200904

Pautasso, C., Zimmermann, O., & Leymann, F. (2008). Restful web services vs. "big"' web services: making the right architectural decision. Proceedings of the 17th International Conference on World Wide Web, 805–814. https://doi.org/10.1145/1367497.1367606

Ramos, R., Preciado-Velasco, J., & Lozano-Rizk, J. E. (2021). Predicción del comportamiento de SLA en redes 5G utilizando inteligencia artificial [Tesis de Maestría, Centro de Educación Científica y Educación Superior de Ensenada, Baja California]. https://cicese.repositorioinstitucional.mx/jspui/bitstream/1007/3656/1/tesis_Rubersy%20Ramos%20Garc%C3%ADa_03%20dic%202021.pdf

Rasheed, I. (2016). *Opportunities in 5G Networks: A Research and Development Perspective*.

RedHat. (2022, June 2). What is an API? https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces#:~:text=API%20stands%20for%20application%20programming,building%20and%20integrating%20application%20software.

Ren, Z., Li, X., Jiang, Q., Wang, Y., Ma, J., & Miao, C. (2023). Network Slicing in 6G: An Authentication Framework for Unattended Terminals. *IEEE Network, 37*(1), 78–86. https://doi.org/10.1109/MNET.112.2100738

Rivas, C., & Mackarena, N. (2022). Sistema distribuido de almacenamiento IPFS [Tesis de licenciatura, Universidad Técnica Federico Santa María]. In 2022. https://hdl.handle.net/11673/54215

Saad, S. Ben, Ksentini, A., & Brik, B. (2021, June 1). A Trust architecture for the SLA management in 5G networks. IEEE International Conference on Communications. https://doi.org/10.1109/ICC42927.2021.9500990

Saian, S. D. S., Sembiring, I., & Manongga, D. H. F. (2024). A Prototype of Decentralized Applications (DApps) Population Management System Based on Blockchain and Smart Contract. JOIV : International Journal on Informatics Visualization, 8(2), 845. https://doi.org/10.62527/JOIV.8.2.1861

Sama, M. R., Beker, S., Kiess, W., & Thakolsri, S. (2016). Service-based slice selection function for 5G. Proceedings - IEEE Global Communications Conference, GLOBECOM. https://doi.org/10.1109/GLOCOM.2016.7842265

Scheid, E. J., Rodrigues, B. B., Granville, L. Z., & Stiller, B. (2019). Enabling Dynamic SLA Compensation Using Blockchain-based Smart Contracts. 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 53–61. https://ieeexplore.ieee.org/document/8717859

Scheid, E. J., & Stiller, B. (2018). Leveraging Smart Contracts for Automatic SLA Compensation-The Case of NFV Environments. In AIMS (pp. 70–74). https://files.ifi.uzh.ch/CSG/staff/scheid/extern/publications/AIMS2018.pdf

Schweizer Zürich, C. (2019). SLAMer: a blockchain-based SLA Management System [Tesis de licenciatura, Universidad de Zurich]. http://www.csg.uzh.ch/

Schwiderowski, J., Pedersen, A. B., & Beck, R. (2024). Crypto tokens and token systems. *Information Systems Frontiers, 26*(1), 319–332. https://doi.org/https://doi.org/10.1007/s10796-023-10382

Solidity. (2024). Solidity Documentation. https://docs.soliditylang.org/en/v0.8.26/

Strehl, B. (2018, September 25). Understanding the honey badger consensus algorithm. Medium; elbstack. https://medium.com/elbstack/understanding-the-honey-badger-consensus-algorithm-240be28d7dc

Suratkar, S., Shirole, M., & Bhirud, S. (2020). Cryptocurrency Wallet: A Review. 4th International Conference on Computer, Communication and Signal Processing, ICCCSP 2020. https://doi.org/10.1109/ICCCSP49186.2020.9315193

Telecommunication Engineering Centre Khurshid Lal Bhawan. (2021). 5G Core Network. https://www.tec.gov.in/public/pdf/Studypaper/5G%20Core%20Network_Study%20Paper_v8.pdf

Tihomirovs, J., & Grabis, J. (2017). Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics. *Information Technology and Management Science, 19*(1). https://doi.org/10.1515/itms-2016-0017

Viriyasitavat, W., & Hoonsopon, D. (2019). Blockchain characteristics and consensus in modern business processes. *Journal of Industrial Information Integration, 13*, 32–39. https://doi.org/10.1016/j.jii.2018.07.004

Visual Studio Code. (2024). Solidity by Nomic Foundation. https://marketplace.visualstudio.com/items?itemName=NomicFoundation.hardhat-solidity

Wan, S., Lin, H., Gan, W., Chen, J., & Yu, P. S. (2023). Web3: The Next Internet Revolution. https://arxiv.org/abs/2304.06111v1

Weihl, W. E. (1990). Remote procedure call. In Distributed Systems (pp. 65–86). Association for Computing Machinery. https://doi.org/10.1145/90417.90740

Wu, W., Zhou, C., Li, M., Wu, H., Zhou, H., Zhang, N., Xuemin, Shen, & Zhuang, W. (2021a). AI-Native Network Slicing for 6G Networks. http://arxiv.org/abs/2105.08576

Wu, W., Zhou, C., Li, M., Wu, H., Zhou, H., Zhang, N., Xuemin, Shen, & Zhuang, W. (2021b). AI-Native Network Slicing for 6G Networks. http://arxiv.org/abs/2105.08576

Xiaoyang, L. (2021, November 4). What is network slicing? how does it work? Https://Info.Support.Huawei.Com/Info-Finder/Encyclopedia/En/Network+Slicing.Html.

Yaga, D., Mell, P., Roby, N., & Scarfone, K. (2018). Blockchain technology overview. https://doi.org/10.6028/NIST.IR.8202

Yang, F., Ding, Z., Yu, Y., & Sun, Y. (2023). Interaction mechanism between blockchain and IPFS. Blockchain, 1(2). https://doi.org/10.55092/BLOCKCHAIN20230007

Yang, M., Li, Y., Li, B., Jin, D., & Chen, S. (2016). Service-oriented 5G network architecture: An end-to-end software defining approach. *International Journal of Communication Systems, 29*(10), 1645–1657. https://doi.org/10.1002/dac.2941

Zanzi, L., Albanese, A., Sciancalepore, V., & Costa-Pérez, X. (2020). NSBchain: A Secure Blockchain Framework for Network Slicing Brokerage. ICC 2020 - 2020 IEEE International Conference on Communications (ICC), 1–7. https://doi.org/10.1109/ICC40277.2020.9149414

Zheng, P., Jiang, Z., Wu, J., & Zheng, Z. (2023). Blockchain-based Decentralized Application: A Survey. https://doi.org/10.1109/OJIM.2022.1234567

Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. Proceedings - 2017 IEEE 6th International Congress on Big Data, BigData Congress 2017, 557–564. https://doi.org/10.1109/BigDataCongress.2017.85

Zheng, Z., Xie, S., Dai, H. N., Chen, W., Chen, X., Weng, J., & Imran, M. (2020). An overview on smart contracts: Challenges, advances and platforms. Future Generation Computer Systems, 105, 475–491. https://doi.org/10.1016/j.future.2019.12.019

Zhou, H., De Laat, C., & Zhao, Z. (2018). Trustworthy cloud service level agreement enforcement with blockchain based smart contract. Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom, 2018-December, 255–260. https://doi.org/10.1109/CloudCom2018.2018.00057

Zhou, S., Li, K., Xiao, L., Cai, J., Liang, W., & Castiglione, A. (2023). A Systematic Review of Consensus Mechanisms in Blockchain. *Mathematics, 11*(10). https://doi.org/10.3390/math11102248