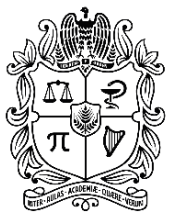


UNIVERSIDAD
NACIONAL
DE COLOMBIA

Prototipo de un Sistema de Energía Transactiva para el aprovechamiento de recursos de energía renovables distribuidos mediante el uso de una tecnología de registro distribuido

Leyla Rocío Becerra Barajas

Universidad Nacional de Colombia

Facultad de Ingeniería

Departamento de Ingeniería de Sistemas y Computación

Bogotá, Colombia

2024

Prototipo de un Sistema de Energía Transactiva para el aprovechamiento de recursos de energía renovables distribuidos mediante el uso de una tecnología de registro distribuido

Leyla Rocío Becerra Barajas

Trabajo final presentado como requisito parcial para optar al título de:

Magister en Ingeniería – Ingeniería de Sistemas y Computación

Codirector:

Ph.D., Jorge Eliécer Camargo Mendoza

Codirector:

Ph.D., Javier Alveiro Rosero García

Línea de Investigación:

Computación aplicada

Grupo de Investigación:

UNSecureLab Research group

Universidad Nacional de Colombia

Facultad de Ingeniería

Departamento de Ingeniería de Sistemas y Computación

Bogotá, Colombia

2024

A mi madre *Martha Leonor Barajas* con amor y admiración por sus esfuerzos, abnegación y enseñanzas durante su paso por esta vida.

A mi hijo *Alejandro*, es mi mensaje de que no debemos rendirnos nunca: los sueños se construyen con dedicación, constancia y amor por lo que se hace.

A Horacio, mi amor y compañero, siempre cómplice y apoyo permanente.

Declaración de obra original

Yo declaro lo siguiente:

He leído el Acuerdo 035 de 2003 del Consejo Académico de la Universidad Nacional. «Reglamento sobre propiedad intelectual» y la Normatividad Nacional relacionada al respeto de los derechos de autor. Esta disertación representa mi trabajo original, excepto donde he reconocido las ideas, las palabras, o materiales de otros autores.

Cuando se han presentado ideas o palabras de otros autores en esta disertación, he realizado su respectivo reconocimiento aplicando correctamente los esquemas de citas y referencias bibliográficas en el estilo requerido.

He obtenido el permiso del autor o editor para incluir cualquier material con derechos de autor (por ejemplo, tablas, figuras, instrumentos de encuesta o grandes porciones de texto).

Por último, he sometido esta disertación a la herramienta de integridad académica, definida por la universidad.

_____

Leyla Rocío Becerra Barajas
Fecha 30/01/2024

Agradecimientos

Me gustaría agradecer a mi familia por toda su paciencia, aliento y apoyo incondicional durante esta y todas las etapas de mi vida.

Agradezco a los profesores Jorge Eliecer Camargo y Javier Rosero por toda su comprensión, su realimentación y los valiosos y oportunos consejos indispensables para la consolidación de este trabajo. Agradezco también cada invitación que recibí de ellos a sesiones de grupos de investigación y participación en eventos académicos: estas experiencias fueron muy importantes por la diversidad de perspectivas a las que me vi expuesta, las cuales me permitieron construir una visión más amplia de la solución requerida al problema identificado.

Agradezco al profesor Jeisson Andrés Vergara Vargas por dedicar tiempo a escucharme, a aconsejarme y por haberme animado a realizar los estudios de maestría.

Agradezco a todos los profesores que durante mi paso por la Universidad Nacional de Colombia bondadosamente compartieron su conocimiento, su tiempo y su trabajo para enriquecer mi visión del mundo y mi formación tanto académica como personal.

Resumen

Prototipo de un Sistema de Energía Transactiva para el aprovechamiento de recursos de energía renovables distribuidos mediante el uso de una tecnología de registro distribuido

Los sistemas de energía transactiva distribuidos se han convertido en mecanismos que favorecen el aprovechamiento de las fuentes de energía renovables al permitir a los nuevos prosumidores comercializar los excedentes de energía dentro de su comunidad. Estos sistemas ofrecen beneficios al habilitar el comercio entre pares. Algunos proyectos actuales han implementado este concepto utilizando tecnologías de registro distribuido (TRD), como Blockchain. Sin embargo, su adopción, especialmente en comunidades pequeñas, implica altos costos de implementación y de operación, largos tiempos de aprobación de transacciones, comisiones en cada transacción y alto consumo de energía. Por lo anterior, este trabajo propone explorar una alternativa tecnológica de registro distribuido que permita la implementación de un prototipo de sistema de energía transactiva distribuida más conveniente para su uso en comunidades locales. Para lograrlo, se identifican las principales características de las TRD y se enumeran las más relevantes. Luego, se describen los aspectos de diseño, desarrollo y pruebas del prototipo de Sistema de Energía Transactiva distribuido, proponiendo su implementación mediante contratos inteligentes y una aplicación descentralizada desplegada de forma independiente usando Ethereum e IoTA como TRD, con el fin de comparar su desempeño. La evaluación entre las dos implementaciones se realiza usando la latencia de las transacciones de escritura como métrica de desempeño, la cual muestra que la TRD IoTA posee menor latencia promedio que Ethereum. La comparación de los resultados obtenidos con un estudio previo presenta nuevos datos que podrían ser utilizados para actualizar, complementar y enriquecer la comprensión y evaluación de estas tecnologías en futuras investigaciones. Finalmente, dadas las características de IoTA se puede concluir que es una tecnología que permite la implementación de sistemas de energía transactiva mediante aplicaciones descentralizadas y contratos inteligentes con mejores tiempos de ejecución que Ethereum. Esto, unido a otras características como el no pago de expensas por aprobación de transacciones y el bajo consumo de energía, ofrece ventajas para su uso en comunidades pequeñas.

Palabras clave: tecnologías de registro distribuido; comercialización de energía; Energía transactiva, P2P, *Blockchain,* recursos renovables, microrredes, comunidades energéticas, Ethereum, IoTA, Contratos Inteligentes, Dapp.

Abstract

Prototype of a Transactive Energy System for the use of distributed renewable energy resources using a Distributed Ledger Technology

Distributed transactive energy systems have become mechanisms that promote the utilization of renewable energy sources by enabling new prosumers to market energy surpluses within their community. These systems offer benefits by facilitating peer-to-peer trading. Distributed transactive energy systems offer benefits by enabling peer-to-peer trading. Some ongoing projects have implemented this concept using distributed ledger technologies (DLT), such as Blockchain. However, their adoption, especially in small communities, entails high implementation and operation costs, delayed transaction approval times, fees for each transaction, and high energy consumption. Therefore, this work proposes exploring an alternative distributed ledger technology that allows the implementation of a more convenient distributed transactive energy system prototype for use in local communities. To achieve this, the main characteristics of DLT are identified and the most relevant ones are listed. Then, the design, development, and testing aspects of the prototype of the distributed Transactive Energy System are described, proposing its implementation through smart contracts and a decentralized application deployed independently using Ethereum and IoTA as DLTs, in order to compare their performance. The evaluation between both implementations is carried out using transaction's writing latency as a performance metric, which shows that IoTA has lower average latency than Ethereum. Comparing the results obtained with a previous study provides new data that could be used to update, complement, and enrich the understanding and evaluation of these technologies in future research. Finally, given the characteristics of IoTA, it can be concluded that it is a technology that allows the implementation of transactive energy systems through decentralized applications and smart contracts with better execution times than Ethereum. This, combined with other features such as no transaction approval fees and low energy consumption, offers advantages for its use in small communities.

Keywords: Distributed Ledger Technologies; energy trading, Transactive Energy, P2P, *Blockchain*, renewable resources, microgrid, energy communities, Ethereum, IoTA, smart contracts, Dapp.

Este Trabajo Final de maestría fue calificado en marzo de 2024 por el siguiente evaluador:

Oscar Julián Perdomo, PhD.
Profesor Facultad de Ingeniería
Universidad Nacional de Colombia

CONTENIDO

Pág.

Lista de figuras

Pág.

Lista de tablas

Pág.

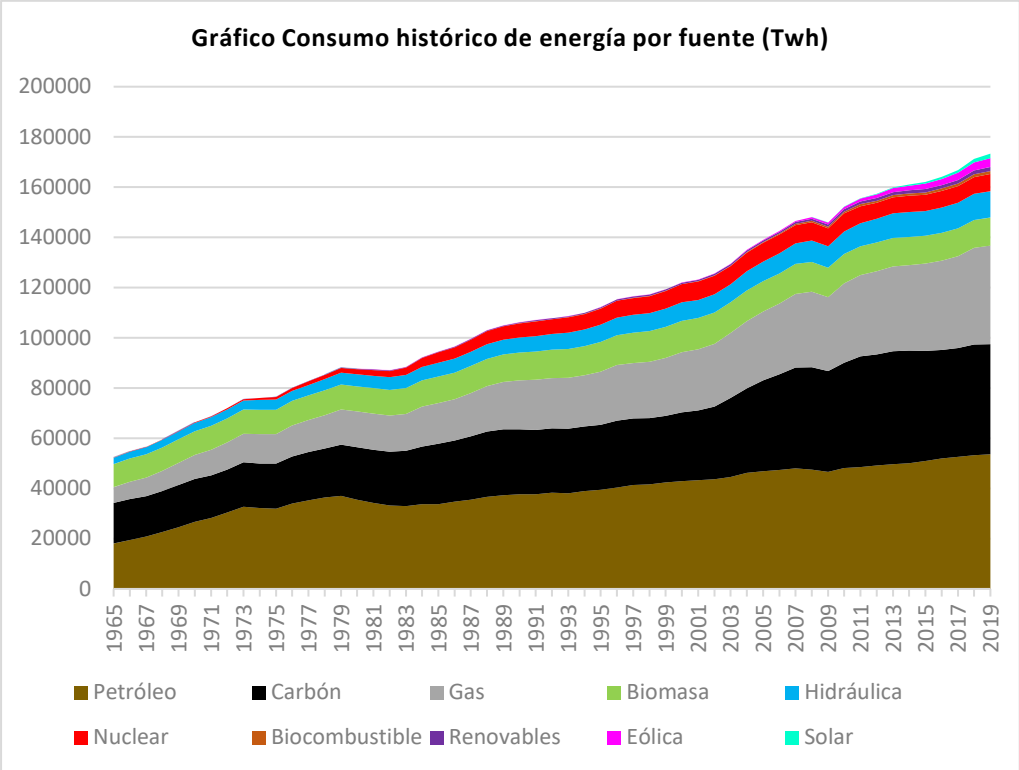
Lista de abreviaturas

Abreviatura	Término
BAF	*Blockchain Applicability Framework*
CDL	*CorDapp Design Language*
DAG	*Directed Acyclic Graph*
Dapp	*Decentralized Application - Aplicación descentralizada*
DER	*Distributed Energy Resources*
DLT	*Distributed Ledger Technology (ies)*
DPOS	*Delegated Proof of Stake*
EVM	*Ethereum Virtual Machine*
FBA	*Federated Byzantine Agreement*
GAD	*Grafo acíclico dirigido*
FER	*Fuentes de energía renovable*
MAB	*Marco de aplicabilidad de Blockchain*
MVE	*Máquina Virtual de Ethereum*
ODS	*Objetivos de desarrollo sostenible*
P2P	*Peer to Peer*
PBFT	*Practical Byzantine fault tolerance*
PoAc	*Proof of activity*
PoAu	*Proof of Authority*
PoB	*Proof of Burn*
PoC	*Proof of Capacity*
PoET	*Proof of Elapsed Time*
PoS	*Proof of Stake*
PoW	*Proof of Work*
RED	*Recursos de energía distribuidos*
TRD	*Tecnología de registro distribuido*

Introducción

El incremento permanente de la población mundial y con él las necesidades de energía eléctrica por parte de los distintos sectores: hogares, industria, transporte, servicios públicos y el comercio, suponen un fuerte requerimiento para aumentar su generación (Ali & Choi, 2020). En la actualidad, la generación de energía está dominada por combustibles fósiles (que son casi el 80% de la energía que se consume, ver **Figura I-0-1**), los cuales son los principales contribuyentes de CO_2 a la atmósfera con un aporte cercano al 73% de las emisiones de gases de efecto invernadero (*ENERGY TRANSITION TOWARDS THE ACHIEVEMENT OF SDG 7 AND NET-ZERO EMISSIONS Secretariat of the High-level Dialogue on Energy 2021 Division for Sustainable Development Goals Department of Economic and Social Affairs*, s/f).

Figura I-0-1 Consumo histórico de energía por fuente.



Elaboración propia con datos de (*Energy Production and Consumption - Our World in Data*, s/f)

Esto plantea un dilema: se requiere generar más energía, pero dados los efectos contaminantes de la mayoría de las fuentes actuales de generación, se requiere así mismo reducir la contaminación asociada a la generación de energía. Una de las soluciones a este dilema, es aumentar el uso de energías renovables y cooperar para el aprovechamiento de tecnologías avanzadas que soporten la implementación de una infraestructura energética basada en energías limpias, como lo propone el objetivo de desarrollo sostenible siete (*Energy - United Nations Sustainable Development*, s/f).

Las fuentes de energía renovables, (conocidas como *FER)*, son instalaciones que aprovechan el potencial energético de la naturaleza: luz solar, el viento, el agua, la energía geotérmica y fuerzas gravitacionales; para suministrar energía eléctrica de forma local (Mengelkamp, Gärttner, et al., 2018). Se debe promover su crecimiento y adopción y para ello la implementación de las redes eléctricas inteligentes y nuevas políticas regulatorias y legales favorables serán las principales facilitadoras (Muhanji et al., 2019). Debido a que las FER no son fuentes de energía constantes pues dependen en gran medida de factores intermitentes de la naturaleza como el clima o los períodos de sol (M. F. Zia et al., s/f) se consideran volátiles. Esta volatilidad hace que su integración a la red de energía global sea limitada, por lo que se convierten en recursos aislados o distribuidos que en la mayoría de los casos no son aprovechados completamente.

Un mecanismo que favorece el aprovechamiento de estas fuentes de energía renovables distribuidas es la comercialización de los excedentes de energía entre los miembros de la comunidad local, es decir, directamente entre pares. Esto genera muchos beneficios: reduce el costo de la energía, fomenta el uso de energías limpias, mantiene las ganancias en la comunidad local, empodera a pequeños productores y promueve comunidades autosostenibles y eficientes (Mengelkamp, Notheisen, et al., 2018). El primer sistema de intercambio de energía entre pares fue propuesto en el proyecto GRIDNET (Skowronski, 2017). Sin embargo, la primera plataforma que permitió la comercialización de energía entre dos vecinos en una microrred fue implementada en Brooklin (Mengelkamp, Gärttner, et al., 2018), y ha servido de base referencial para otras publicaciones y proyectos a nivel mundial.

La tecnología más usada en el comercio de energía entre pares es *Blockchain,* sobre la cual existen a nivel mundial más de 122 organizaciones dedicadas a la investigación de su uso en el sector de energía y al menos 40 proyectos de energía transactiva distribuida (Mengelkamp, Gärttner, et al., 2018). La principal razón usar *Blockchain* es el grado de madurez de esta tecnología y la diversidad de aplicaciones ya implementadas que prueban sus beneficios (Andoni et al., 2019). Sin embargo, su uso, en especial en pequeñas comunidades, implica altos costos de implementación, elevados requerimientos computacionales y consumo energético asociado a ellos, dependencia de la volatilidad del Bitcoin, largos tiempos de aprobación de transacciones, comisiones en cada transacción y mineros validadores que preferirían procesar transacciones más grandes (Cullen et al., 2020). Dadas estas implicaciones, es importante estudiar y probar tecnologías de registro distribuido alternas que permitan solventar las limitantes técnicas y económicas que *Blockchain* podría tener para la implementación de un sistema de intercambio de energía.

Este trabajo explora los conceptos básicos y características de las tecnologías de registro distribuido que tienen implicaciones en el diseño e implementación de un prototipo de sistema de energía transactiva distribuida que soporte el aprovechamiento de recursos de energía renovables en una comunidad local. Para ello, se propusieron los siguientes objetivos:

Objetivo general

El objetivo general de este trabajo es desarrollar un prototipo de un sistema de energía transactiva para el aprovechamiento de recursos de energía renovables distribuidos mediante el uso de una tecnología de registro distribuido.

Objetivos específicos

Los objetivos específicos que redundan en el cumplimiento del objetivo general son:

1. Determinar los aspectos de diseño del prototipo de sistema de energía transactiva distribuida por medio del análisis de los requerimientos del sistema y de las características funcionales de la tecnología de registro distribuido seleccionada.

2. Implementar el prototipo de sistema de energía transactiva distribuida mediante su codificación y prueba.
3. Evaluar el desempeño del prototipo de sistema de energía transactiva en un ambiente simulado.

Contribución

El principal aporte de este trabajo fue la implementación de un prototipo de sistema de energía transactiva a través de un Gateway que integra la información de consumos y producción proveniente de los medidores de energía de la microrred en una aplicación descentralizada para la compra/venta de energía entre pares mediante tokens fungibles. Para lograrlo, se desarrollaron dos contratos inteligentes que fueron desplegados en dos tecnologías de registro distribuido: Ethereum e *IoTA*. Al final, se pudo comparar el rendimiento de esta aplicación en cada una de estas dos tecnologías mediante un mecanismo propuesto para medir la latencia de ejecución transacciones de escritura.

Los contratos inteligentes, el código de la aplicación Gateway y de las aplicaciones asociadas a cada tecnología de registro distribuido, se han dispuesto en respectivos repositorios públicos con el fin de que puedan ser utilizados por otras personas como recurso para replicar los hallazgos de este trabajo o como base para iniciar una nueva implementación o estudio de este tipo de aplicaciones. Los enlaces a estos repositorios están incluidos en el Anexo 2.

Finalmente, como un aporte adicional se encuentra en preparación un artículo que permitirá divulgar entre la comunidad científica el trabajo realizado:

Becerra, L. Camargo, J. Rosero, J. (2024) *Transactive energy prototype for energy communities using a distributed ledger technology.*

Estructura del documento

Este documento se encuentra organizado en tres capítulos principales:

El primer capítulo presenta los conceptos claves de microrredes, sistemas de energía transactiva y tecnologías de registro distribuido, los cuales son necesarios para el entendimiento de los requerimientos y distintos aspectos de diseño de una aplicación de energía transactiva distribuida.

En el segundo capítulo se presentan el análisis de requerimientos, arquitectura de implementación, y el modelamiento usado para el diseño del prototipo del Sistema de Energía Transactiva para el aprovechamiento de recursos de energía renovables distribuidos. Así mismo describe la forma como se implementaron y probaron cada uno de los módulos planteados en el diseño.

El tercer capítulo describe la métrica y metodología seleccionada para evaluar el desempeño del prototipo. Dado que el prototipo se desplegó en dos tecnologías de registro distribuido, se expone la comparación entre las mismas.

Por último, se presentan las conclusiones del trabajo realizado, algunas propuestas para trabajos futuros, las referencias bibliográficas que sirvieron de base a este trabajo y se incluyen algunos anexos que permitirán documentar lo expuesto en el cuerpo del documento.

1. Marco conceptual

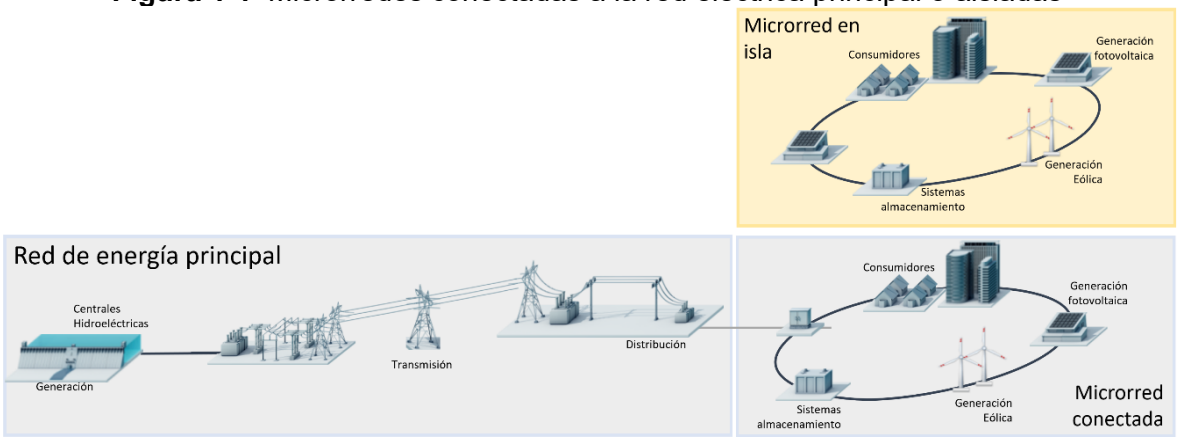
En este capítulo se plantean los principales conceptos, definiciones y características de los recursos de energía distribuidos, los sistemas de energía transactiva y las tecnologías de registro distribuido que fundamentan los requerimientos, el diseño y la implementación del *prototipo de un* sistema de energía transactiva propuesto en este trabajo.

1.1. Recursos de energía distribuidos

Los recursos de energía renovables por naturaleza no están concentrados en un área específica: son recursos de energía distribuidos (RED) que generalmente se aprovechan de forma local y cuya integración a la red de energía es limitada puesto que su alta volatilidad pone en riesgo la confiabilidad de la misma. Es la nueva red eléctrica inteligente la que permitirá el aprovechamiento del potencial de las microrredes. La *red eléctrica inteligente* se define como un sistema de energía que permite la comunicación y el flujo de energía bidireccional a través funciones de control y toma de decisiones avanzadas (Muhanji et al., 2019).

Una microrred es un dominio específico de las *redes eléctricas inteligentes* que ha sido definida por (Marnay et al., 2015) como “un sistema de distribución de energía que contiene cargas y recursos de energía distribuidos (RED) que se operan de forma controlada y coordinada, y que pueden estar conectados o aislados de la red eléctrica principal” (ver **Figura 1-1**). Una microrred, también puede definirse como una comunidad en la que se agrupan consumidores, generadores y *prosumidores* de energía, los cuales constituyen un mercado local en el cual se comercializa energía (Vieira & Zhang, 2021). Los *prosumidores* de energía son un nuevo concepto que ha evolucionado con la digitalización de los RED (M. F. M. F. Zia et al., 2019). Los prosumidores son usuarios que producen y consumen energía eléctrica y por tanto pueden participar en un mercado de energía local para vender los excedentes de su generación de energía una vez han suplido su consumo particular. Los RED y los prosumidores están cambiando la cadena de valor de la energía (M. F. Zia et al., s/f). Los mercados de energía en microrredes brindan a los prosumidores y consumidores una plataforma de mercado para comercializar dentro de su comunidad la energía generada localmente (Mengelkamp, Gärttner, et al., 2018).

Figura 1-1 Microrredes conectadas a la red eléctrica principal o aisladas



Elaboración propia

1.2. Sistemas de energía transactiva

Los mecanismos que facilitan la integración de los prosumidores para el intercambio de energía son conocidos como sistemas de energía transactiva (M. F. Zia et al., s/f). La energía transactiva ha sido definida por el *GridWise® Architecture Council* (Transactive Energy Systems Research, Development and Deployment Roadmap Prepared by the GridWise ® Architecture Council, 2018) como “un sistema de mecanismos económicos y de control que permite el equilibrio dinámico de la oferta y la demanda en toda la infraestructura eléctrica utilizando el valor como parámetro operativo clave”. En (Muhanji et al., 2019) la energía transactiva es definida como el “conjunto de técnicas para gestionar el intercambio de energía mediante transacciones comerciales”.

Un mercado de energía debe contar con un sistema de información de alto rendimiento que permita la conexión de todos los participantes del mercado, proporcionando la plataforma, el acceso y la supervisión de las operaciones del mercado (Mengelkamp, Gärttner, et al., 2018). Para su implementación, se requiere un sistema de información eficiente, escalable y confiable con una resolución de adecuada. Debe implementarse de manera que todos los participantes del mercado tengan el mismo acceso para evitar discriminación.

Además, de acuerdo con (Siano et al., 2019) las características mínimas que debe garantizar un sistema de energía transactiva deberían ser:

- Seguridad de los datos, debido a que se intercambian datos financieros.
- Privacidad de los datos para evitar la perfilación de los usuarios.
- Velocidad de ejecución de transacciones de orden lineal o menor.
- Resiliencia a fallos.
- Integridad de la información.
- Huella energética reducida.

Los sistemas de energía transactiva pueden ser implementados de forma centralizada o descentralizada. Cada uno de ellos representa ciertas ventajas como se puede ver en la **Tabla 1-1**.

Tabla 1-1 Cuadro comparativo entre tipos sistemas de energía transactiva

Característica	Sistema centralizado	Sistema descentralizado
Escalabilidad	Bajo	Alto
Transparencia	Bajo	Alto
Confiabilidad	Bajo	Alto
Costo computacional	Alto	Bajo
Costo de comunicaciones	Alto	Bajo
Centrado en el usuario	Bajo	Alto
Dificultad de implementación	Bajo	Alto

Fuente (M. F. M. F. Zia et al., 2019)

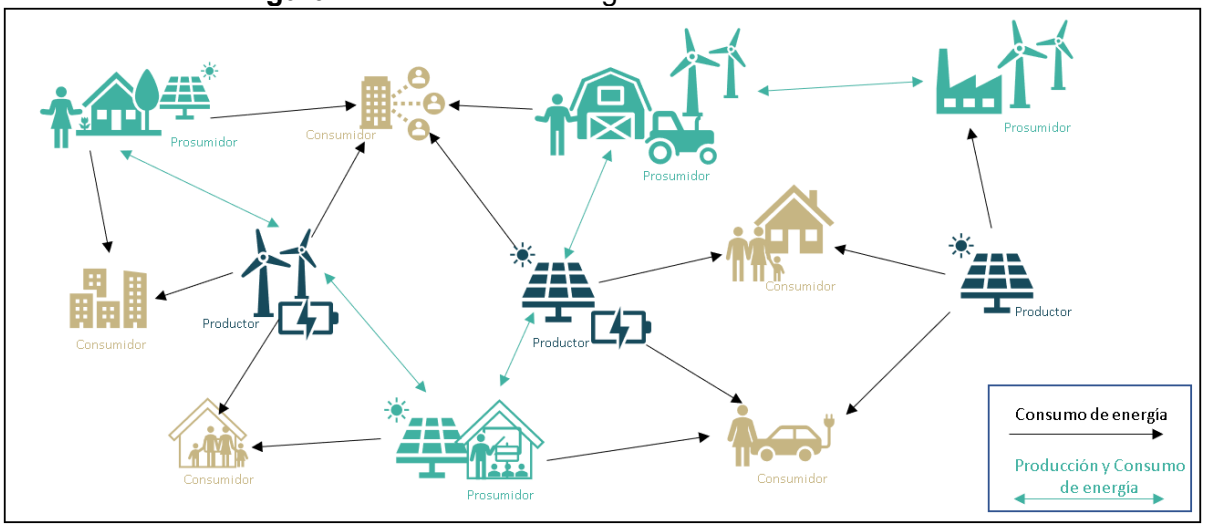
En la actualidad los sistemas de energía transactiva tradicionales son en su mayoría gestionados de forma centralizada por una única autoridad la cual es responsable de controlar las operaciones, conocida como el *operador* de la microrred (Siano et al., 2019). Todos los participantes del mercado le envían sus ofertas y demandas de energía utilizando la misma infraestructura. El operador es el responsable de la estructura técnica y operativa de la microrred y de determinar el precio de la energía por medio de la consolidación de la demanda/oferta del mercado (M. F. Zia et al., s/f). Estos sistemas se ven expuestos a indisponibilidad por fallas, ataques cibernéticos y amenazas de seguridad puesto que toda la infraestructura y los datos se encuentran en una única plataforma central. El costo de implementación de este sistema es mayor por los altos requerimientos

computacionales y de infraestructura de comunicaciones para atender grandes cantidades de solicitudes por parte de los participantes. La escalabilidad de estos sistemas en la medida que aumentan los prosumidores participantes es vertical lo que implica mayores recursos económicos y estrategias más complejas de redundancia. Finalmente, el manejo de la información y los costos de la energía centradas en el operador de la microrred le resta transparencia y equidad en el mercado de cara a los demás participantes (M. F. Zia et al., s/f).

Existen diversos mecanismos de implementación de sistemas de energía transactiva que están diseñados para permitir una operación más descentralizada del sistema eléctrico, lo cual redunda en una mejor utilización de los activos de la red y una mejor integración de los recursos energéticos distribuidos a través del balance energético local (Hayes et al., 2020). En estos sistemas de energía transactiva no existe un operador único de la microrred. Los prosumidores participan sin depender de una única fuente de control. Existen sistemas parcialmente distribuidos o descentralizados en los que algunos de los participantes toman un rol de intermediarios (como *brókeres* y agregadores) en el mercado y participan del control de la operación y cálculo del valor de la energía (Siano et al., 2019). En los sistemas completamente distribuidos todos los nodos contribuyen al control del valor de la energía, a la ejecución y aprobación de las transacciones en un ambiente seguro y transparente (M. F. Zia et al., s/f). Para la aprobación de las transacciones se usa un mecanismo de consenso que permite la validación y almacenamiento de nuevos datos. Debido a lo anterior, se reduce el riesgo de que un ciberataque a un nodo ponga en riesgo el sistema entero.

El uso de tecnologías de registro distribuido ha sido propuesto como habilitador de sistemas de energía transactiva distribuida para que generadores, prosumidores y consumidores locales puedan realizar sus operaciones comerciales en un entorno virtual seguro facilitado por la protección de la información mediante el uso de hashes criptográficos (Siano et al., 2019) como es mostrado en la **Figura 1-2**.

Figura 1-2 Sistema de energía transactiva distribuido



Elaboración propia

1.3. Tecnologías de registro distribuido

Las tecnologías de registro distribuido TRD (DLT por sus siglas en inglés) proporcionan una infraestructura para desarrollar aplicaciones descentralizadas sin una autoridad central, en las cuales se pude registrar, compartir y sincronizar transacciones en activos digitales (Antal et al., 2021). Existen diferentes tipos de implementaciones de TRD, la mayoría de las cuales han surgido para abordar problemas específicos en aplicaciones descentralizadas. Dado que los registros son públicos y se encuentran replicados, brindan una gran transparencia garantizando confidencialidad mediante el uso de mecanismos criptográficos de llave pública. Las estructuras de datos con mecanismos de *hash* permiten una trazabilidad fácil de los activos y actualizaciones del registro a prueba de manipulaciones.

En (Górski & Bednarski, 2020)y (Jabed Morshed Chowdhury et al., s/f) se enumeran las características de las TRD:

1. Capacidad para lograr un consenso sobre el estado del registro sin depender de un ente centralizado de confianza.
2. Inmutabilidad e irreversibilidad del estado del registro.
3. Persistencia de las transacciones en cada uno de los nodos participantes

4. No repudio de los datos: uso de firmas digitales que utilizan claves criptográficas que garantizan la autenticidad de la fuente de datos.
5. Control distribuido de los datos, lo cual garantiza alta tolerancia a fallos.
6. Auditoría y transparencia.

La base de las TRD son las redes *Peer to Peer o P2P,* en las cuales cada dispositivo puede ser tanto un proveedor como un receptor de recursos y puede comunicarse directamente con el resto sin la mediación de un nodo intermediario. La comercialización de energía entre prosumidores ha sido comparada con lo que ocurre en las redes *P2P* en (Giotitsas et al., 2015), por lo cual se usa este concepto para dar soporte a principios de economía colaborativa como estructura que facilita el intercambio de bienes entre pares (Sousa et al., 2019) y que se conoce como *comercialización de energía P2P o* sistema *"peer-to-peer"* P2P de comercio de energía.

1.3.1. Mecanismos de consenso

En las TRD, se requiere un mecanismo para generar y posteriormente aceptar una estructura de datos. Una vez se propone un bloque de datos, el paso siguiente es que los miembros de la red acepten el bloque propuesto mediante un proceso llamado *consenso*.

Los protocolos de consenso son un conjunto de algoritmos y datos estructurados que deben tener cierta medida de resiliencia en caso de falla de los nodos, retrasos o corrupción de los mensajes (Miglani et al., 2020) (bien sea por fallas en la comunicación o por alteraciones maliciosas). Por lo cual cumplen las siguientes propiedades (Dr & Baliga, 2020):

- Confiabilidad: todos los nodos participantes producen los mismos resultados de acuerdo con las reglas del protocolo.
- Actividad: todos los nodos activos del sistema producirán un valor.
- Tolerancia a fallos: el protocolo de consenso puede seguir ejecutándose, aunque exista la falla de un nodo debido a errores considerados benignos (como fallas de hardware, comunicaciones o de software), o debido a fallas Bizantinas que pueden ocurrir por errores de software que generan el funcionamiento errático del nodo (Dr & Baliga, 2020).

De acuerdo con (Xu et al., 2017), los mecanismos de consenso incluyen las reglas para la validación y emisión de transacciones y bloqueos, resolución de conflictos y el esquema de incentivos. El consenso permite la adición de nuevas transacciones, que cada transacción sea válida y agregada una sola vez.

Existen numerosos mecanismos de consenso (Zheng et al., 2017), (Andoni et al., 2019), como se pueden ver en la
Tabla 1-2.

Tabla 1-2 Mecanismos de consenso

Mecanismo	Descripción	Retos
PoW: Proof of Work	En este mecanismo los nodos de la red validadores son conocidos como mineros, quienes compiten para resolver un reto criptográfico de generación de un hash que empieza con cierto número consecutivo de ceros en las posiciones más significativas. Es un proceso de cálculo por fuerza bruta que requiere un esfuerzo computacional (Andoni et al., 2019).	El principal reto de este mecanismo es el consumo de recursos de energía eléctrica. Adicionalmente los nodos mineros deben hacer grandes inversiones en infraestructura para poder competir.
PoS: Proof of Stake	En este mecanismo, que fue propuesto como alternativa más eficiente a PoW, se reemplaza el esfuerzo computacional por un proceso de selección aleatoria en la cual la probabilidad de ser seleccionado para determinar el consenso es proporcional a la riqueza del validador (Andoni et al., 2019).	La selección basada en la riqueza del validador incrementa la posibilidad de que el nodo más rico de la red termine dominándola elección (Zheng et al., 2017).
PBFT: Practical Byzantine fault tolerance	Mediante este mecanismo, todos los nodos de la red votan, mediante un proceso que tiene tres pasos: antepreparación, preparación y ejecución. en una ronda. En cada ronda se seleccionan nodos clasificados a la siguiente ronda, mediante una votación de 2/3 de todos los nodos. Se requiere que cada nodo sea conocido en la red, para que su prestigio apoye su elección (Zheng et al., 2017) (Andoni et al., 2019).	Tiene retos en términos de seguridad, pues algunos nodos maliciosos podrían ponerse de acuerdo en las rondas para validar ciertos bloques.
DPOS: Delegated Proof of Stake	En este mecanismo, cada nodo de la red elige unos delegados y testigos, que son los que finalmente validan los bloques. Está basado en la reputación de los nodos (Andoni et al., 2019).	Los retos están del lado de una posible centralización causada por la baja participación de los nodos en los procesos de elección (Andoni et al., 2019).
FBA: Federated Byzantine Agreement	En este mecanismo, cada nodo confía en un pequeño grupo de validadores que considera confiables, una vez que estos validadores han aprobado transacciones, los miembros las aceptan en un proceso que tiene varias rondas hasta lograr la aprobación en un consenso del 80% del total de votos.	Tiene retos en términos de seguridad, pues algunos nodos maliciosos podrían ponerse de acuerdo en las rondas para validar ciertos bloques.

Mecanismo	Descripción	Retos
PoAu: Proof of Authority	En este mecanismo se otorga autoridad a ciertos nodos para hacer cambios a la cadena de bloques o autorizaciones. Un bloque es aceptado si ha recibido la firma de los nodos autorizados.	Posible centralización.
PoET: Proof of Elapsed Time	Algoritmo propuesto por Intel, en el que se utilizan nuevas funcionalidades de CPU y un entorno de ejecución confiable. El validador solicita un tiempo de espera confiable. El nodo con el tiempo de espera más corto produce el bloque.	Dependencia de un fabricante: Intel
PoAc: Proof of activity	Este algoritmo combina PoW y PoS, los nodos mineros generan las plantillas de bloques vacías y luego son validadas por un conjunto aleatorio seleccionado en función de su riqueza.	Combina las desventajas de PoW y PoS: desperdicio de recursos y problemas con validadores doble firma.
PoB: Proof of Burn	En este algoritmo los nodos de validación pagan el privilegio de validar bloques con dinero que se "quema" y no se puede volver a utilizar.	Desperdicio innecesario de recursos económicos y de energía.
PoC: Proof of Capacity	En este algoritmo se requiere que los validadores asignen espacio en disco duro para aumentar sus posibilidades de producir el siguiente bloque y obtener su recompensa.	Algún validador puede decidir crear dos bloques y firmarlos porque no tiene nada que perder.

Elaboración propia basada en (Miglani et al., 2020) ,(Zheng et al., 2017) y (Andoni et al., 2019).

La metodología utilizada para llegar a un consenso determina en gran medida características clave de rendimiento como la escalabilidad, la velocidad de transacción, la finalidad de la transacción, la seguridad y el gasto de recursos como la electricidad (Andoni et al., 2019).

1.3.2. Clasificación de las TRD

De acuerdo con los dominios de aplicación se pueden tener distintos tipos de TRD (M. F. Zia et al., s/f):

Públicas: Cualquier usuario puede acceder a la red en cualquier momento. Puede presentar problemas con la privacidad de la información.

Privadas: Se debe obtener una membresía para acceder a la red. Es ideal para aplicaciones que requieren garantizar la privacidad de los datos.

Así mismo, entre ellas y de acuerdo con la autorización para verificación y validación de transacciones se clasifican en

Con restricciones o *Permissionless*: en estas TRD sólo las entidades autorizadas y consideradas confiables pueden participar en el registro de las transacciones.

Sin restricciones o *Permissioned*: en estas TRD cualquier nodo participante puede crear y validar transacciones, modificar el registro y actualizar datos.

El acceso a lectura y escritura sobre la cadena de bloques/transacciones de acuerdo con el tipo de TRD es mostrado en la **Tabla 1-3**.

Tabla 1-3 Cuadro comparativo entre tipos de *Blockchain*

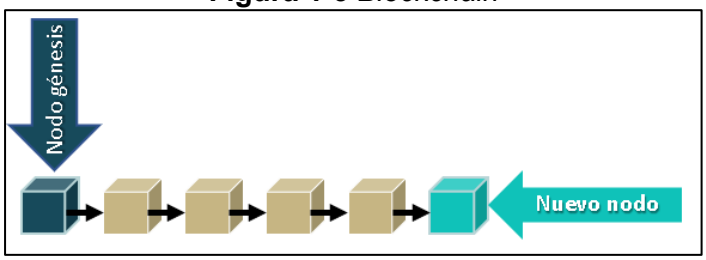
Acción	Pública		Privada	
	Con restricciones	**Sin restricciones**	**Consorcio**	**Empresarial**
Acceso a la cadena	Todos	Todos	Propietario	Propietario
Acceso a las transacciones	Todos	Propietarios y usuarios validados	Propietarios y usuarios validados	Administrador
Aprobar bloques y adicionarlos a la cadena	Todos	Propietarios y usuarios validados autorizados	Propietarios y usuarios validados autorizados	Administrador

Elaboración propia basada en (Antal et al., 2021)

1.3.3. *Blockchain*

La TRD más popular y ampliamente utilizada en diversas aplicaciones es conocida como *Blockchain.* En ella los mensajes son almacenados en forma de bloques de transacciones válidas, cada uno de los cuales contiene metainformaciones relativas a otro bloque de la cadena anterior en una línea de tiempo (Zheng et al., 2017). En *Blockchain,* el registro digital de transacciones es generado y modificado por todos los participantes de una red P2P a través de un protocolo seguro y compartido en la red distribuida. En esta red, los nodos conocidos como mineros ejecutan algoritmos para evaluar, verificar y hacer coincidir la información de la transacción con el historial de transacciones de *Blockchain.* Un nuevo bloque se aprueba solo si la mayoría de los nodos está a favor de la transacción. El nuevo bloque, una vez aprobado, se agrega a la cadena existente (Bertone et al., 2020), ver **Figura 1-3**.

Figura 1-3 *Blockchain*



Elaboración propia

La cadena de bloques reúne todos los beneficios de las funciones *hash* criptográficas y con la integración de un algoritmo de consenso, asegura un registro histórico inmutable de toda la actividad de la red (Antal et al., 2021). Sin embargo, *Blockchain* con el aumento de la cantidad de transacciones presenta problemas de uso del ancho de banda, retrasos en la aprobación de las transacciones y altas tarifas por transacción (M. F. Zia et al., s/f).

1.3.4. Ethereum

Propuesto como la segunda era del *Blockchain*, *Ethereum* habilita la posibilidad de programar aplicaciones descentralizadas con funciones, formatos y reglas de transición de estado arbitrarias usando la última capa abstracta de la cadena de bloques. En su primera versión utilizó el algoritmo de consenso PoW, pero éste fue actualizado en la versión 2.0 a PoS (*Proof-of-stake (PoS)* | *ethereum.org*, s/f). Estas funciones son programadas a través de cajas criptográficas que contienen valor y solo lo desbloquean si se cumplen ciertas condiciones, conocidos como *contratos inteligentes* (Buterin, 2015).

Los contratos inteligentes, codifican las condiciones de acuerdos entre los participantes y se ejecutan automáticamente casi en tiempo real cuando las mismas se cumplen. Cuando el contrato es desplegado, se genera una única dirección y cuando esta dirección recibe una transacción, el código se activa permitiendo leer o escribir en el almacenamiento interno y así ejecutar diversas funciones (Geun Song et al., 2021).

Todos los nodos en una red P2P pueden ejecutar contratos inteligentes a través de la ejecución de una transacción con el número de dirección del contrato.

Actualmente existen diversos lenguajes de programación que permiten la codificación de contratos inteligentes, el más usado de ellos es *Solidity*.

Un contrato inteligente está conformado por un balance de cuenta, un código de programa y un archivo de almacenamiento. El código de programa del contrato inteligente se compila y su *bytecode* es almacenado en la *Blockchain* y ejecutado por la Máquina Virtual de *Ethereum* MVE (Ethereum Virtual Machine o EVM por sus siglas en inglés). Una vez el contrato inteligente es desplegado, su código no puede ser modificado. El código del programa es invocado cada vez que se recibe un mensaje.

Ethereum, representa una alternativa interesante para la implementación de un sistema de energía transactiva, pues permite la implementación de cadenas de bloque privadas que se pueden editar e implementar para adaptarse a diferentes escenarios (Chen et al., 2021).

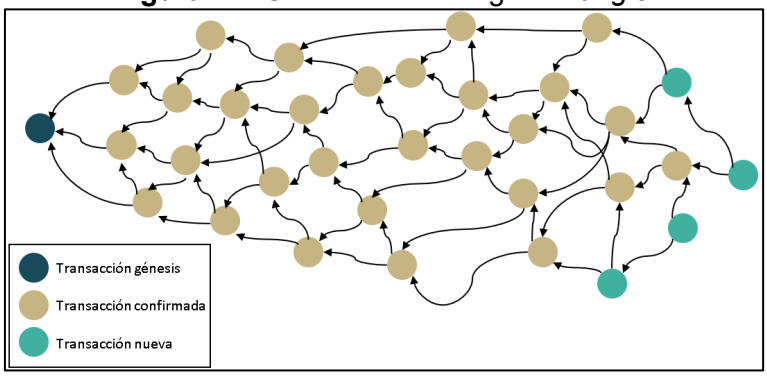
1.3.5. TRD de Grafo acíclico dirigido

La Estructura de Grafo Acíclico Dirigido GAD o *DAG* (por sus siglas en inglés *Directed Acyclic Graph*) es una TRD que ha sido denominada la versión 3 de *Blockchain* (Dong et al., 2019). Existen varias propuestas de plataforma basadas en variaciones GAD como *Next*, *IoTA*, *Orumesh*, *Nano*, *Byteball, XDAG* y *Dagcoin* entre otras (Pervez et al., 2019). Consiste en una estructura de datos que está compuesta por transacciones individuales vinculadas en forma de grafo acíclico dirigido que valida transacciones usando métodos de orden topológico.

En *IoTA*, la TRD con estructura GAD más usada, el grafo inicia con una transacción génesis que es aprobada por todas las transacciones en el grafo. La aprobación de una nueva transacción está sujeta a un algoritmo basado en un método de Cadenas de Markov Monte Carlo (Antal et al., 2021). Cada participante en la red debe aprobar dos transacciones anteriores para emitir una nueva transacción. Así, en la medida que ingresan nuevas transacciones se validan más transacciones; lo que origina una red distribuida de transacciones con doble verificación (Pervez et al., 2019). Debido a que no se requiere que mineros autoricen cada transacción y que es suficiente con que dos nodos la validen, el proceso se ejecuta de forma más ágil. Este proceso ocurre en una red subyacente denominada *Tangle* (Dong et al., 2019), cuya representación gráfica es representada en la **Figura 1-4.** *Tangle* también puede ser descrita como una base de

datos distribuida y sin comisiones que permite transacciones fuera de línea (Bertone et al., 2020).

Figura 1-4 Grafo acíclico dirigido: *Tangle*



Elaboración propia

1.3.6. Comparación entre algunas TRD

Algunos autores han abordado el análisis comparativo de las TRD, cada uno ellos han incluido distintas TRD en su estudio: en (Antal et al., 2021) se presenta un cuadro comparativo de las principales características de las TRD *Blockchain* y GAD y se incluye un tercer tipo el cual es un híbrido entre los dos anteriores. En (Bertone et al., 2020) se incluyen dos modelos adicionales: *Hashgraph* y *Sidechain*. En (M. F. Zia et al., s/f), se incluyen los modelos *Holochain* y *Tempo*. En (Jabed Morshed Chowdhury et al., s/f) se realizado una comparación cualitativa de distintas TRD considerando los algunos parámetros que pueden determinar su viabilidad de uso (ver **Tabla 1-4**,): tipo, tamaño del bloque, tiempo medio de ejecución de una transacción (o cálculo del bloque), si tiene costo por validación de transacciones, el consumo de energía y el algoritmo de consenso. De acuerdo con este cuadro comparativo, algunas de las plataformas ofrecen características de interés para la implementación de un sistema de energía transactiva distribuido: tamaño de la transacción o del bloque ligero, latencia reducida, sin costos o pagos asociados a la aprobación de la transacción, bajo o muy bajo consumo de energía y un algoritmo de consenso que garantice el equilibrio entre participantes mientras protege la red de nodos maliciosos que quieran influir en la aprobación de las transacciones.

Tabla 1-4 Tabla de comparación cualitativa entre algunas TRD

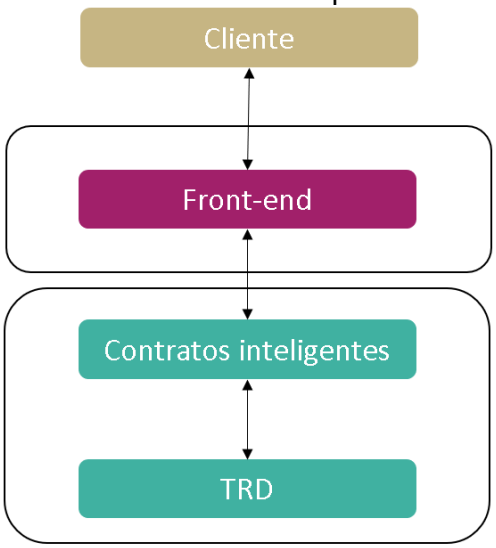
Plataforma	Tipo	Escalabilidad		Costo	Consumo de Energía	Algoritmo de consenso
		Tamaño del bloque	Tiempo transacción			
Bitcoin	Pública	1MB	10m	En Bitcoins	Alto	PoW
Ethereum	Pública	Restringido	15s	En Ether	Alto	V1 PoW V2 PoS
EOS	Pública	1MB, configurable dinámicamente	0.5s	Si	Sin información	DPoS
Cardano	Pública	Aprox. 1 MB	20s	Si	No	Ouroboros
Fabric	Privada	Configurable	0.5s-2s	No	Muy bajo	Orderers-Endorsers
Sawtooth	Privada	Configurable	NA	No	Muy bajo	PoET
IoTA	Privada	Configurable	Configurable	No hay pagos	Muy bajo	Tangle
Multichain	Privada	Configurable	Configurable	Si (versión empresarial)	Muy bajo	Consenso distribuido y validadores
Corda	Privada	Configurable	0.5s-2s	No hay pagos	Muy bajo	Notarios
WaltonChain	Pública y Privada	225 TX	30s	Si	Bajo	WPoC

Elaboración propia inspirada en (Jabed Morshed Chowdhury et al., s/f)

1.4. Aplicaciones descentralizadas

Las aplicaciones descentralizadas, conocidas como Dapp, son programas que permiten registrar, compartir y sincronizar transacciones en activos digitales sin una autoridad central (Antal et al., 2021) mediante el uso de una TRD. Generalmente una aplicación descentralizada consta de una interfaz de usuario (*front-end*) que emite llamadas directas a una infraestructura *back-end* descentralizada compuesta por los contratos inteligentes y la TRD (Wohrer et al., 2021) como se representa en la **Figura 1-5.**

Figura 1-5 Estructura básica de una aplicación descentralizada



Elaboración propia basada en (Wohrer et al., 2021)

Generalmente una aplicación descentralizada está compuesta por algunos componentes conceptuales comunes (Wohrer et al., 2021), entre ellos:

1.4.1. Billetera

En una aplicación descentralizada, los usuarios no usan una cuenta protegida por una clave almacenada en un servidor centralizado. En realidad, usan una clave pública, la cual es distribuida abiertamente como identificación de su cuenta y una clave privada que únicamente conoce el usuario y que debe protegerse. Debido a la complejidad nemotécnica que generalmente tiene la llave privada, ésta se almacena en una Billetera. La billetera puede ser un dispositivo físico, un programa o un servicio que se encarga de cifrar, firmar y reenviar transacciones a la TRD. Ver (1) en **Figura 1-6**.

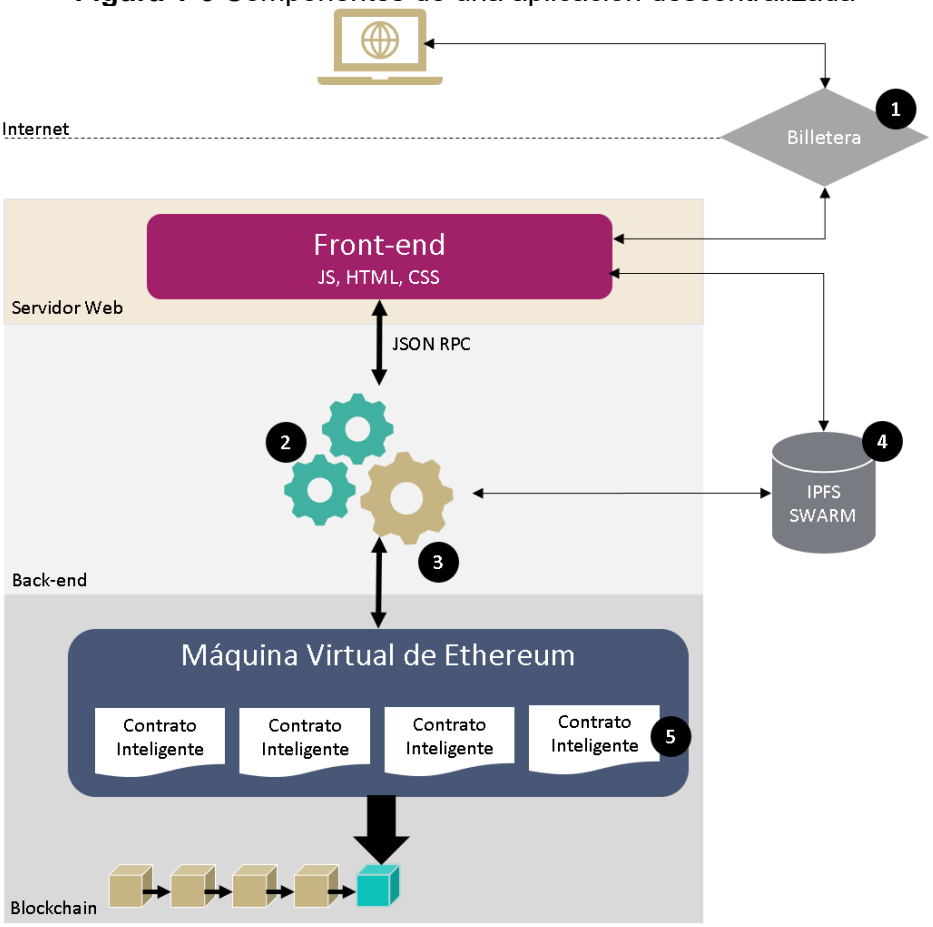
1.4.2. Servicios perimetrales

Se trata de componentes que están disponibles de forma pública en internet y que pueden brindar diversos servicios como autenticación, contenido, seguridad, *proxies* inversos, balanceo de carga o interfaces de aplicación, entre otros. Ver (2) en **Figura 1-6**.

1.4.3. Gestión de Identidad y acceso

Este componente gestiona la información del usuario y respalda los procesos de autenticación y autorización, con lo cual se tiene control sobre el acceso a los recursos y aplicaciones. En este tipo de aplicaciones es importante identificar los requerimientos sobre la confidencialidad de la identidad dentro de la TRD.

Figura 1-6 Componentes de una aplicación descentralizada



Elaboración propia basada en (Wohrer et al., 2021) y (*The Architecture of a Web 3.0 application*, s/f)

1.4.4. Lógica del back-end

Una aplicación bien sea monolítica o una suite de microservicios organizados, que implementan la lógica requerida para el logro de los objetivos del negocio. Ver (3) en **Figura 1-6**.

1.4.5. Almacenamiento fuera de la cadena

Consiste en cualquier mecanismo de almacenamiento que fuera de la cadena de bloques o TRD que contenga datos relacionados con la cadena de bloques. Generalmente se seleccionan los datos a ser almacenados fuera de la cadena cuando se requieran consultas rápidas, cuando sean de gran tamaño o requieran ser modificados. Ver (4) en **Figura 1-6**.

1.4.6. Nodo en la cadena de bloques

Un nodo en la cadena de bloques es un dispositivo que ejecuta un programa que implementa el protocolo *Blockchain* (Wohrer et al., 2021). Un nodo completo tiene almacenada y disponible toda la cadena de bloques. Un nodo ligero contine solo los encabezados de los bloques y puede validar transacciones con el soporte de un nodo completo.

1.4.7. Máquina virtual de Ethereum

La Máquina Virtual de Ethereum (MVE o EVM por sus siglas en inglés *Ethereum Virtual Machine*) es un entorno completo y aislado en el que se ha implementado una máquina completa de Turing que permite la ejecución de contratos inteligentes en un ambiente seguro.

1.4.8. Contrato inteligente

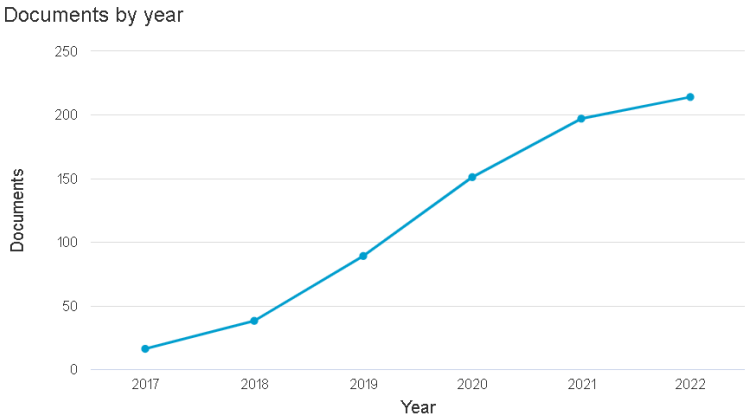
Un contrato inteligente es una aplicación sencilla que implementa en la cadena de bloques la lógica del negocio. Una vez desplegado, el contrato es inmodificable, lo que asegura que la aplicación se comporte como se espera y que no haya manipulación en la ejecución del programa. Ver (5) en **Figura 1-6**.

1.5. Estudios y proyectos de energía transactiva P2P

La tendencia creciente de publicaciones asociadas a aplicaciones de energía transactiva entre pares mostrada en la **Figura 1-7** evidencia el interés de la comunidad académica por el uso de tecnologías con alto potencial disruptivo como *Blockchain* y sus aplicaciones al sector energético. Además, un análisis de coocurrencia de palabras en la entre los artículos analizados durante este trabajo, revela que los términos más usados son

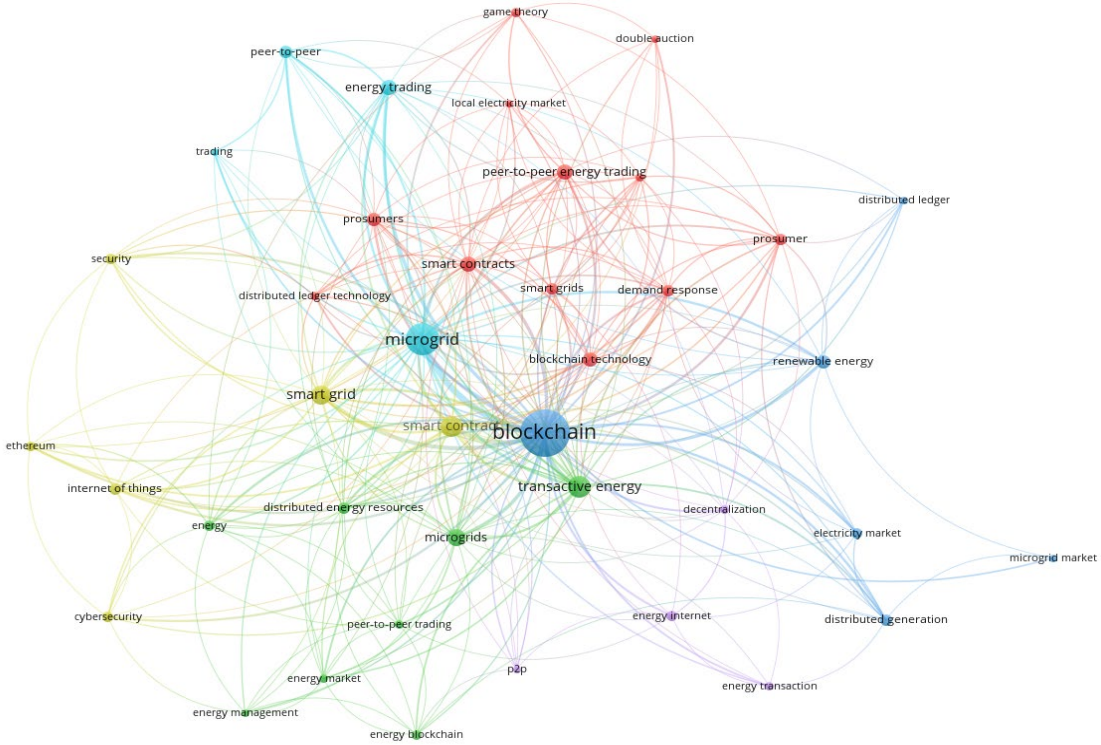
"Blockchain", "microgrid", "transactive energy" y "Smart contracts" y están altamente correlacionados entre ellos (ver en **Figura 1-8** Red de coocurrencia de palabras).

Figura 1-7 Publicaciones de energía transactiva en redes P2P por año



Fuente Base de datos SCOPUS (Elsevier, 2022)

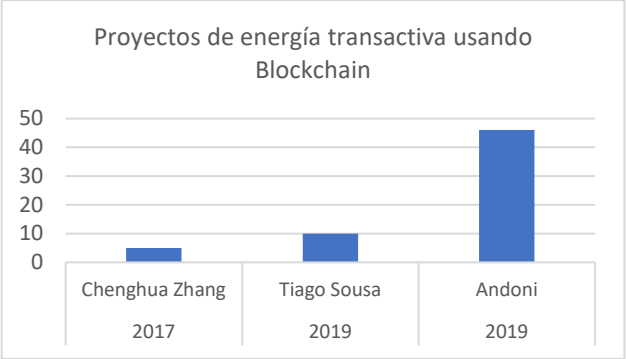
Figura 1-8 Red de coocurrencia de palabras



Elaboración propia usando *VOSViewer*

Este crecimiento de proyectos y de plataformas puede ser evidenciado en la **Figura 1-9**. en donde la tendencia muestra la cantidad de plataformas que fueron identificadas y categorizadas por (Zhang et al., 2017), (Sousa et al., 2019) y (Andoni et al., 2019) (todos usan tecnología *Blockchain*).

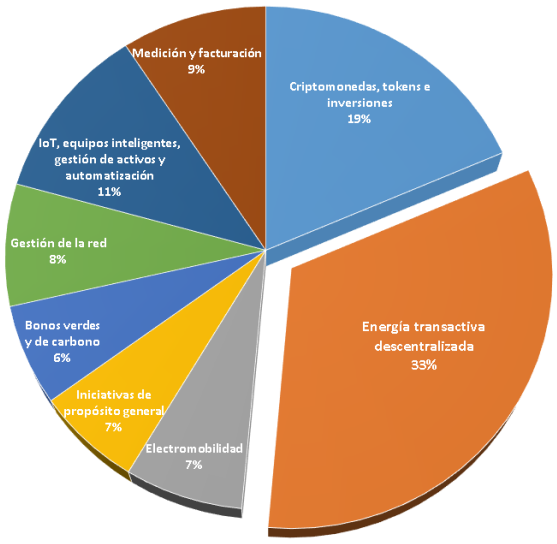
Figura 1-9 Cantidad de proyectos de energía transactiva con *Blockchain*



Elaboración propia

Los datos recopilados por (Andoni et al., 2019), hasta el año 2019, permiten evidenciar varios aspectos: se identificaron 140 proyectos de aplicaciones de *Blockchain* en el sector de energía, de los cuales 46 proyectos corresponden a sistemas de energía transactiva descentralizada para RED, es decir el 33%, ver **Figura 1-10**.

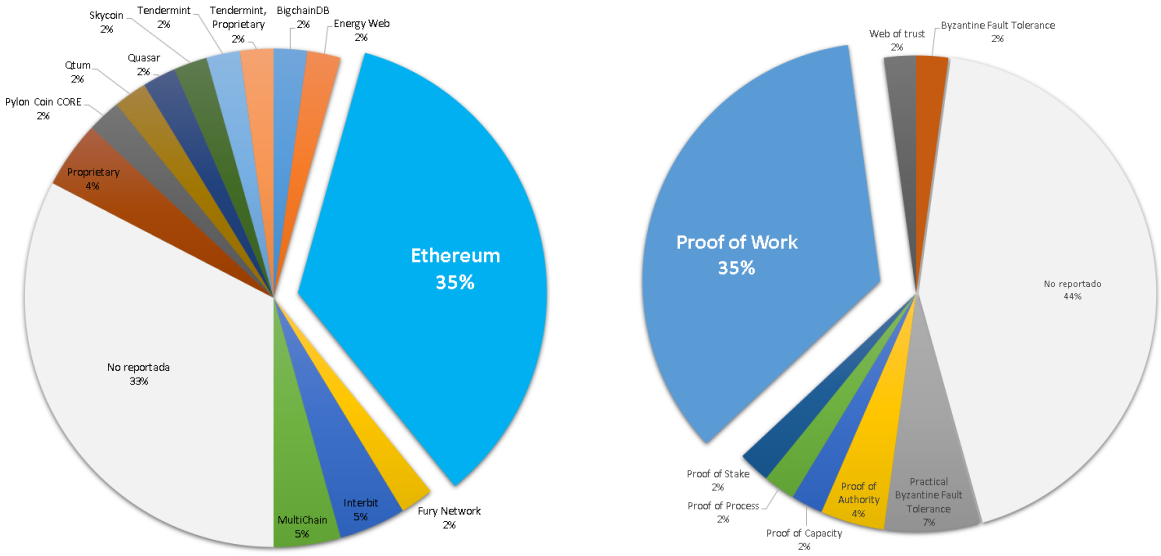
Figura 1-10 Aplicaciones de *Blockchain* en el sector energético



Fuente (Andoni et al., 2019)

De acuerdo con los datos mostrados en la **Figura 1-11**, de los proyectos de energía transactiva descentralizada identificados por (Andoni et al., 2019), al menos el 35% de ellos como plataforma *Ethereum*, es decir son aplicaciones descentralizadas que aprovechan el uso de contratos inteligentes para su implementación. Además, se evidencia que un 35% usan como algoritmo de consenso Proof of Work, sin embargo, un 44% no reporta el mecanismo de consenso utilizado.

Figura 1-11 Proyectos de aplicaciones de energía transactiva descentralizada

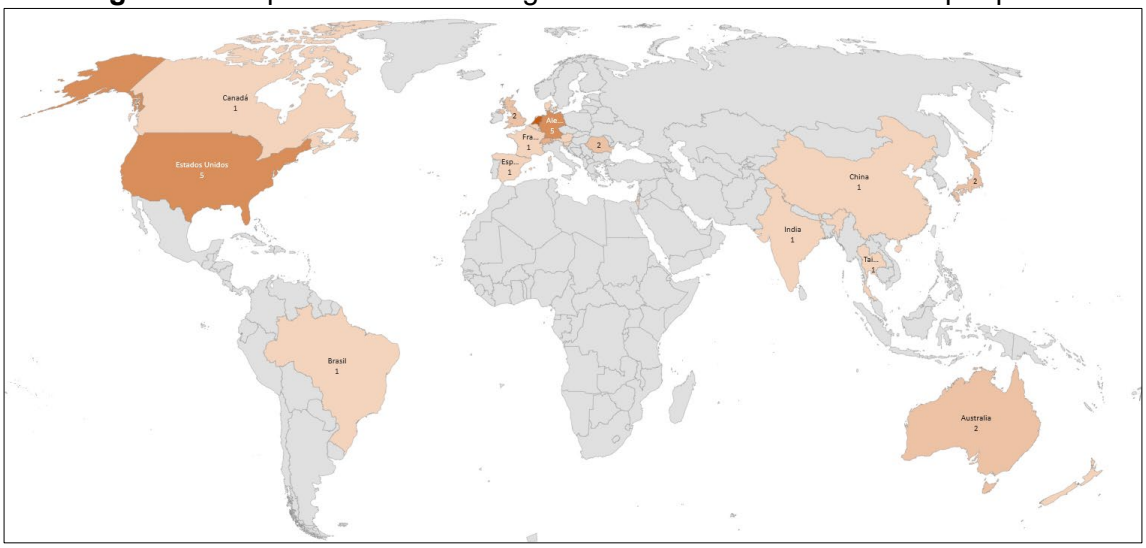


Elaboración propia basada en datos de (Andoni et al., 2019)

Los países con mayor cantidad de implementaciones son Holanda (8), Alemania (5), Estados Unidos (5), Suiza (3) y Singapur (3). Ver **Figura 1-12**.

Los aspectos discutidos en este capítulo permiten la caracterización de las tecnologías de registro distribuido, que son la base fundamental para el diseño del prototipo de sistema de energía transactiva distribuida propuesto en este trabajo. Consideraciones como el tipo de TRD, el mecanismo de consenso y la capacidad para ejecutar contratos inteligentes serán aspectos importantes a tener en cuenta en los criterios de diseño y selección de las tecnologías de registro distribuido más apropiadas para la implementación del prototipo. Estos aspectos se abordarán en algunos de los apartados del siguiente capítulo: Prototipo de software.

Figura 1-12 Aplicaciones de Energía Transactiva descentralizada por país.



Elaboración propia basada en datos de (Andoni et al., 2019)

2. Prototipo de Software

El objetivo principal de este trabajo es la implementación de un prototipo de sistema de energía transactiva mediante el uso de una tecnología de registro distribuido que permita habilitar el comercio de energía entre productores, consumidores y prosumidores de una comunidad. Este capítulo aborda los aspectos de diseño e implementación del prototipo: requisitos, arquitectura o modelo topológico, módulos a implementar para el cumplimiento de los requerimientos y mecanismos y patrones de codificación.

2.1. Análisis de requisitos

El proceso de diseño del prototipo se inició con la identificación de los requisitos que el sistema de energía transactiva debe cumplir. Algunos de ellos fueron extraídos de la propuesta de (Kirpes et al., 2019). Los requisitos identificados para la implementación de este prototipo son de tipos funcional y no funcional:

2.2. Requisitos funcionales

Estos requisitos consisten en declaraciones que permiten determinar la forma como el sistema se debe comportar para satisfacer las necesidades de los usuarios. Para el prototipo de sistema de energía transactiva se definieron los requisitos que se enumeran en la **Tabla 2-1**.

Tabla 2-1 Requisitos funcionales del prototipo

Requisito	Descripción
RF1.	Debe permitir el intercambio económico en una moneda que pueda ser usada con facilidad por la comunidad
RF2.	Debe permitir la integración de la información de la red de energía en cuanto a consumos, producción y energía disponible en el mercado.
RF3.	Debe permitir la integración de distintos agentes del mercado: consumidores, productores, prosumidores
RF4.	Debe permitir al usuario consumidor conocer su consumo.
RF5.	Debe permitir al usuario productor monitorear su generación

RF6.	Debe permitir al usuario prosumidor monitorear su consumo y su producción.
RF7.	Debe permitir a los usuarios prosumidores y generadores establecer la tarifa por kWh
RF8.	Debe permitir la comercialización de energía entre pares

2.3. Requisitos no funcionales

Los requisitos no funcionales son los criterios que se tendrán en cuenta para garantizar aspectos de seguridad, rendimiento, escalabilidad y estabilidad del sistema, entre otros. Para el prototipo de sistema de energía transactiva se definieron los requisitos que se enumeran en la **Tabla 2-2**.

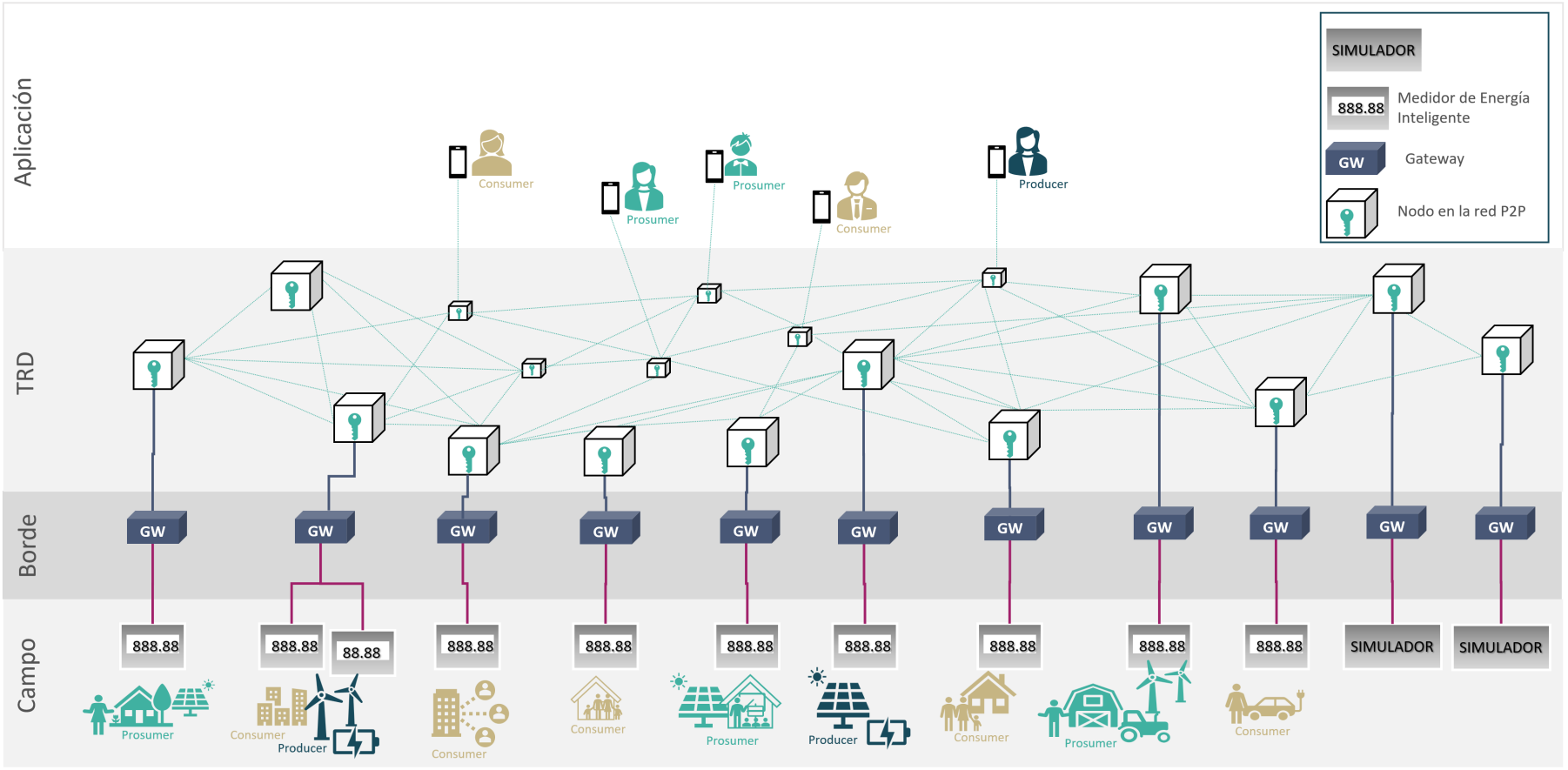
Tabla 2-2 Requisitos no funcionales del prototipo

Requisito	Descripción
RNF1.	Debe garantizar privacidad de los datos de los participantes del mercado
RNF2.	Debe garantizar la seguridad, integridad y confidencialidad de las transacciones ejecutadas por los participantes del mercado
RNF3.	Debe ser eficiente: la velocidad de ejecución de transacciones de orden lineal o menor.
RNF4.	Debe utilizar modelo de datos canónicos estandarizados para la lectura de medidores
RNF5.	Debe poderse integrar con la infraestructura física disponible
RNF6.	Debe ser resiliente a fallos
RNF7.	Debe tener una huella energética reducida
RNF9.	Debe ser escalable
RNF10.	Debe garantizar transparencia entre los participantes del mercado
RNF11.	Debe ser descentralizada y no depender de un único ente.

2.4. Arquitectura del sistema:

La arquitectura del sistema se ha estructurado en un modelo de capas de acuerdo con la **Figura 2-1**.

Figura 2-1 Arquitectura del sistema



Elaboración propia

2.4.1. Capa de campo

El objetivo de esta capa es garantizar la infraestructura necesaria para la distribución de energía dentro de la microrred. En ella están incluidos los medidores inteligentes de energía, dispositivos de control e interruptores que permitan la activación/desactivación de los puntos de intercambio de energía. Para la implementación de un sistema de energía transactiva descentralizado, los medidores de energía deben cumplir al menos las siguientes características:

1. Medición bidireccional de energía
2. Entradas binarias que permitan la supervisión del estado del interruptor de integración a la microrred.
3. Salidas binarias que permitan la activación del interruptor para intercambio de energía
4. Puerto de comunicaciones con protocolo estándar para facilitar la integración a los nodos de la TRD.

2.4.2. Capa de Borde

El objetivo de la capa de borde es servir de límite entre los dispositivos de campo y la red P2P de la TRD.

2.4.3. Capa de TRD

Esta capa es la responsable de la implementación de la TRD. En ella se encuentran los nodos interconectados en una red P2P, que les permite la ejecución de la TRD seleccionada. Es la capa responsable de la ejecución distribuida del algoritmo de consenso y los mecanismos de almacenamiento de los datos.

Esta capa aprovecha las características de las TRD para garantizar la seguridad, integridad, confidencialidad y no repudio de las transacciones gracias al uso de funciones criptográficas.

2.4.4. Capa de Aplicación

Esta capa que estará en contacto con el usuario final. Es la responsable de ofrecer una interfaz que permita al usuario conocer la información sobre su uso de energía (consumo, producción, compras) así como el estado del mercado en general.

2.5. Diseño del prototipo

2.5.1. Criterios de diseño

Los criterios de diseño usados tuvieron en cuenta las características de cada una de las capas de la arquitectura descrita en el capítulo 2.4.

2.5.1.1. Capa de campo

Se consideraron los siguientes mecanismos que permiten la implementación y prueba de la funcionalidad de reporte de datos de energía desde la capa de campo:

Integración del medidor de energía inteligente

El dispositivo responsable de medir y desplegar la información de consumo y producción de energía de los consumidores, productores y prosumidores es el medidor de energía inteligente. Generalmente estos dispositivos cuentan con protocolos estándares de comunicación que les permite integrarse a los sistemas de información para el envío de información de forma periódica. Uno de los protocolos más generalizados entre estos dispositivos es el protocolo MODBUS el cual puede ser usado en redes seriales de campo o redes TCP/IP. El uso difundido de dicho protocolo es la razón por la cual se ha seleccionado como mecanismo de integración al prototipo, pues permitirá ampliar y diversificar las pruebas y aplicaciones a un número mayor de dispositivos de distintos fabricantes del mercado.

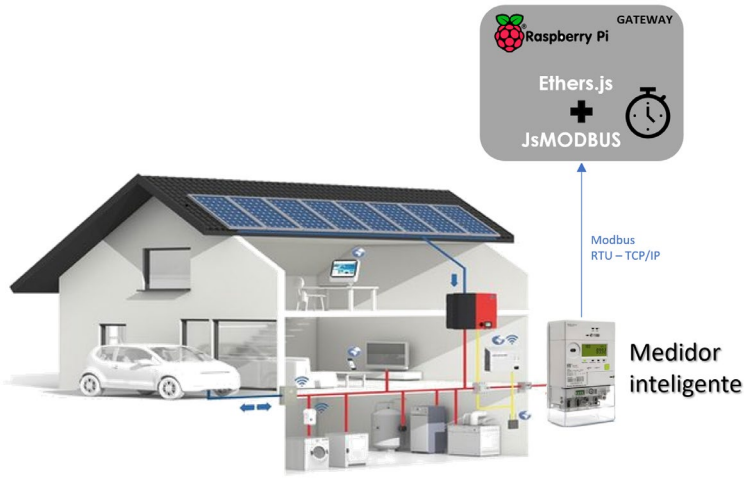
Simulación de medidor de energía

Con el objetivo de poder escalar las pruebas, se implementa un módulo que enviará valores de energía simulados al Gateway, los cuales serán transmitidos y escritos en la TRD. De esta forma es posible ejecutar pruebas simuladas con varios “medidores” sin necesidad de realizar la conexión real con los mismos.

2.5.1.2. Capa de borde

En la capa de borde se implementa un *Gateway* que permite la obtención de los datos desde la capa de campo mediante telegramas MODBUS RTU o TCP/IP o mediante un simulador. El *Gateway* consiste en una aplicación sencilla que realiza lecturas periódicas de los datos y las convierte en transacciones que son enviadas a la red P2P de la TRD. La implementación de esta integración es mostrada en la **Figura 2-2**.

Figura 2-2 Integración del medidor mediante un Gateway virtual



Elaboración propia

2.5.1.3. Capa de TRD

En la capa de la TRD, se deben tener en cuanto varios criterios de diseño con el fin de seleccionar la TRD apropiada para la aplicación del prototipo de sistema de energía transactiva:

1. Debe permitir la ejecución de funciones a través del uso de contratos inteligentes los cuales permitirán la implementación de la aplicación descentralizada mediante las librerías y billeteras digitales ya probadas en otras aplicaciones.
2. Debe disponer de un mecanismo de consenso resiliente a fallos, con una velocidad de ejecución de orden lineal y con huella energética reducida con el fin de garantizar tiempos de respuesta apropiados para el reporte permanente de datos de energía desde la capa de campo.

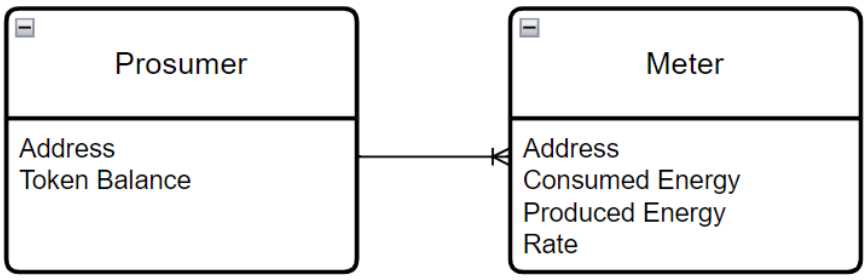
2.5.1.4. Capa de Aplicación

En esta capa se determinó como criterio de diseño la selección de una interfaz que pueda ser fácilmente ejecutada en cualquier navegador, para facilitar el acceso a los usuarios y para aprovechar las extensiones de billetera digital existentes. De esta manera la aplicación puede hacer uso de patrones de código que ya han sido probados y validados en otras aplicaciones.

2.5.2. Modelo de datos

Se realizó un modelo de datos que representa la relación entre las entidades principales del sistema de energía transactiva. Con el fin de unificar los distintos roles de un usuario en el mercado: consumidor, productor y prosumidor, se decidió etiquetarlos como con el nombre *Prosumer*. Se considera que un prosumidor debe tener al menos un medidor (*meter)* para poder participar en el mercado y vender sus excedentes de energía. Este modelo de datos general se muestra en la figura **Figura 2-3**.

Figura 2-3 Modelo de datos



Elaboración propia

2.5.3. Contratos inteligentes

La estrategia para la implementación del sistema de energía transactiva fue el uso de una TRD que pudiera desplegar contratos inteligentes para poder habilitar el desarrollo de la aplicación descentralizada.

Para la creación del (los) contrato(s) inteligente(s) se decide abordar el problema desde dos puntos de vista:

- La información que proviene de campo a través de la lectura de un medidor. Información que permite totalizar la energía consumida y producida por cada nodo asociado a un prosumidor
- Los mecanismos de intercambio económico que se utilizarán para la comercialización de la energía entre prosumidores.

Como consecuencia de lo anterior, se determina la creación de dos contratos inteligentes:

2.5.3.1. Contrato de la Microrred

El mecanismo por medio del cual se consolida la información de energía consumida y producida procedente de cada medidor inteligente participante en la microrred. El contrato permite establecer la relación de propiedad entre el prosumidor y el medidor o medidores a su cargo.

Con el fin de garantizar los requisitos no funcionales asociados a la protección de la confidencialidad del prosumidor y de impedir la perfilación de sus consumos o generación de energía, se ha asignado a cada una de estas entidades una dirección criptográfica única, la cual puede ser creada mediante la billetera digital de acceso a la TRD: el campo ***Address***. Esta dirección le servirá para para firmar sus transacciones y para almacenar pequeñas cantidades de dinero que le permitirán el pago de expensas. Así mismo, ofrece otras ventajas, puesto que permite al prosumidor:

- Crear las direcciones de los medidores y gestionarlas desde su billetera digital
- Otorgarles permisos de uso de una cantidad limitada de fondos a las direcciones asignadas a los medidores (que podrán ser usados para el pago de sus transacciones).
- Designar un número al medidor que no pueda ser fácilmente correlacionado con el prosumidor por los demás integrantes del mercado, con lo cual se protege la privacidad del mismo.

2.5.3.2. Contrato del mercado

Para la gestión del mercado se decidió utilizar el estándar ERC20 para tokens fungibles sobre la TRD *Ethereum,* el cual fue implementado y es mantenido por *Openzeppelin*

(*OpenZeppelin | Contracts*, s/f). Los contratos que heredan las características del estándar ERC20 permiten la creación de un token de intercambio comercial el cual tiene nombre propio y una tasa de transferencia a Ethereum. Seleccionar este estándar facilita la implementación del mercado mientras ofrece mayor seguridad e interoperabilidad con otras aplicaciones gracias a reglas, auditorías y validaciones permanentes de la comunidad de desarrolladores que lo soporta. Así mismo, permite el cumplimiento de los requerimientos en cuanto al uso de una moneda consolidada y de fácil intercambio comercial para los participantes del mercado.

2.5.3.3. Uso del patrón de diseño Factory

El patrón de diseño *Factory Contract*, el cual consiste en la creación de una plantilla de contrato inteligente que puede generar instancias de contrato inteligente a partir del modelo integrado en su código fuente (*Factory Contract – Blockchain Patterns*, s/f) fue utilizado para desplegar múltiples instancias del mercado, para que sean utilizadas por microrredes distintas: con cada creación de un nuevo mercado local personalizado mediante parámetros se realiza el despliegue automático de un contrato asociado a una nueva microrred.

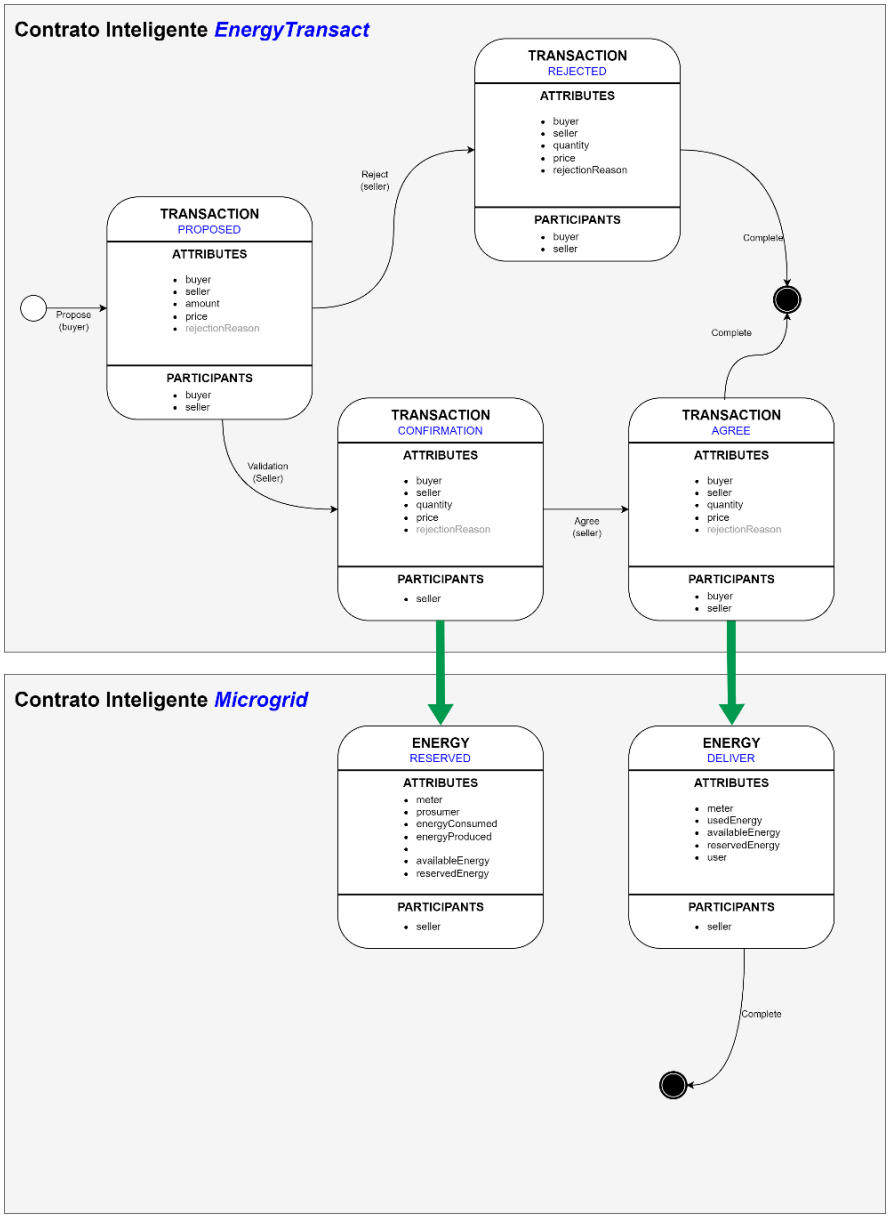
2.5.3.4. Integración de los contratos

La integración entre el contrato de la microrred y el del mercado de energía es indispensable puesto que se requiere suplir de información sobre la energía disponible en la microrred al mercado, así como de gestionar y documentar el intercambio de energía a nivel de la microrred cuando se haya realizado una transacción económica en el mercado.

El proceso por medio del cual se realiza la compra de energía requiere una interrelación entre los dos contratos la cual ha sido desplegada en la ver

Figura 2-4 usando CDL (*CorDapp Design Language (CDL) overview - R3 Documentation*, s/f).

Figura 2-4 Procesos interrelacionados entre los contratos



1. El comprador realiza una solicitud de compra confirmando al vendedor la cantidad de energía que desea adquirir. Se pre-aprueba una transferencia por el valor de esta energía.
2. Desde el contrato del mercado de energía ocurre una validación de la disponibilidad de la energía para ser vendida. En caso de no estar disponible, la transacción es rechazada. En caso de estar disponible la energía es reservada hasta que se haga efectiva la transacción económica.

3. Es enviada una transacción económica preaprobada por el monto requerido para pagar la cantidad de energía de acuerdo con la tarifa de NRG/kwh
4. Si la transacción económica se ejecuta exitosamente (en el contrato del mercado de energía), la energía reservada (en el contrato de la microrred) pasa a estado de entregada y se actualizan todos los datos del prosumidor y del mercado
5. En caso de que la transacción económica no llegue a buen término, la energía que fue reservada vuelve a estar disponible.

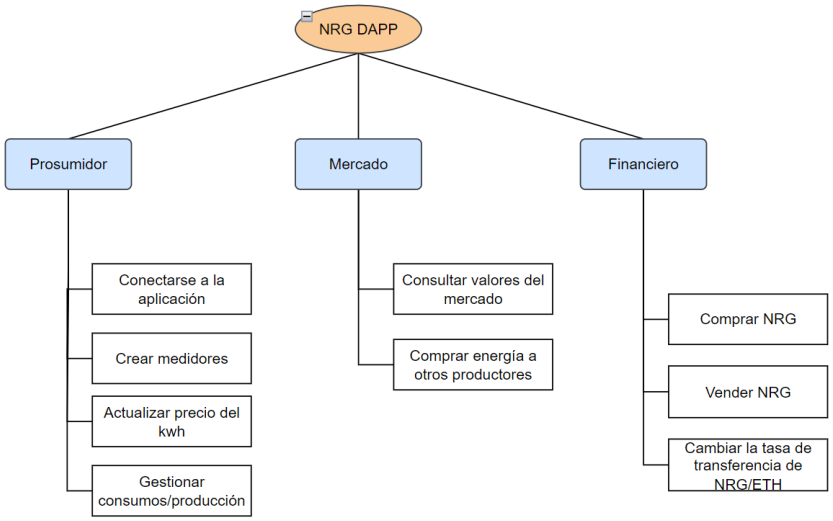
2.5.4. Aplicación descentralizada

La aplicación descentralizada es la responsable, de ofrecer al usuario final (consumidores, productores, prosumidores de energía) las funcionalidades básicas del mercado las cuales le permiten el intercambio comercial de energía en la moneda o token de la plataforma. Como mecanismo de intercambio se ha definido una moneda con un nombre referencial: NRG. De acuerdo con los requerimientos del prototipo, las funcionalidades disponibles son:

1. Actualización de la tasa de intercambio entre la moneda del mercado y la moneda comercial (por ejemplo, Ethereum)
2. Compra y venta de NRG para ser usados como mecanismo de intercambio en la compra de energía
3. Comercializar tokens NRG por ETH para que las ganancias puedan ser monetizados en otras plataformas del ecosistema de criptomonedas.
4. Compra de energía a otros productores o prosumidores del mercado
5. Creación de nuevos medidores de energía por parte del prosumidor
6. Actualización de las tarifas de energía para cada medidor

El diseño de la aplicación contempla esta funcionalidad mediante despliegues alineados de acuerdo con el tipo de información: particular del mercado de energía (prosumidor), general del mercado de energía (mercado) y gestión de tokens e intercambios con la moneda base (financiero). En la **Figura 2-5** se despliega la descomposición funcional asociada.

Figura 2-5 Descomposición funcional de la aplicación descentralizada



Elaboración propia

2.6. Codificación y despliegue del prototipo

El prototipo fue implementado de acuerdo con el diagrama mostrado en la **Figura 2-6** el cual permite mostrar las interrelaciones entre las distintas instancias de código los módulos funcionales y la arquitectura de la aplicación.

2.6.1. Selección de TRD

De acuerdo con los criterios de diseño planteados en el capítulo 2.5.1, se evalúan las TRD con mecanismos de implementación de contratos inteligentes. Se decide seleccionar Ethereum como TRD base para el prototipo dado su grado de madurez y diversidad de aplicaciones en las que ha sido probada. Con el fin de poder realizar un análisis comparativo con otra TRD, se aprovecha la implementación de *IoTA* para contratos inteligentes usando la MVE. Para cada caso, se decide realizar la implementación en la red de pruebas de la respectiva TRD. Las características principales de las TRD seleccionadas son mostradas en la **Tabla 2-3**.

Cada una de estas redes de prueba cuentan con un proveedor que permite asignar recursos en su moneda a las cuentas de los usuarios de prueba. Con lo cual no es necesario realizar una inversión económica muy alta para la implementación y evaluación del prototipo.

Tabla 2-3 Comparativo de características de las TRD seleccionadas

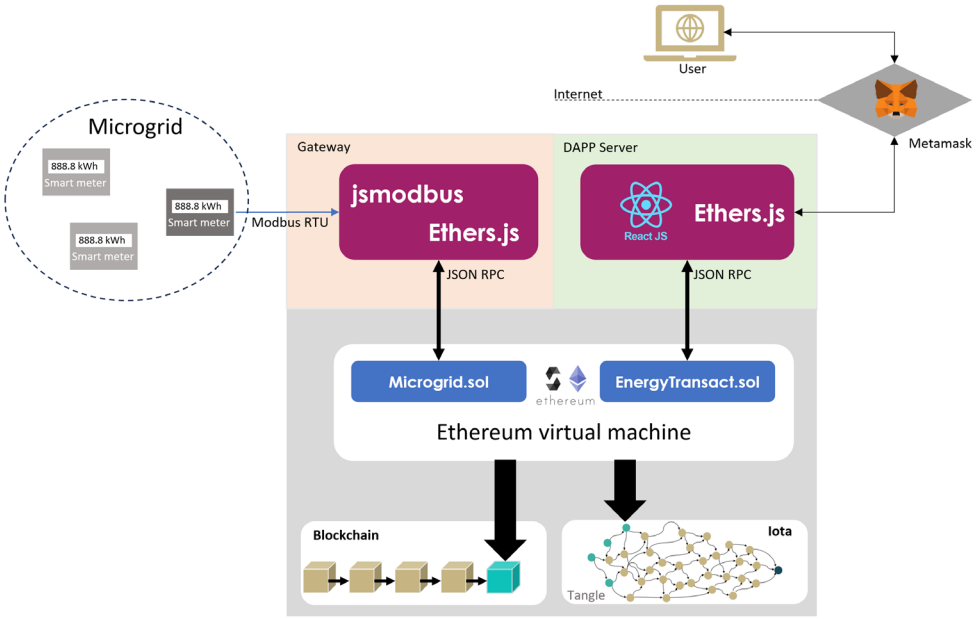
Característica	Ethereum	*IoTA*
Red de pruebas	Sepolia	Shimmer
Algoritmo de consenso	PoS: Proof of Stake	PoW: Proof of Work
Validadores	Grupo cerrado limitado	Grupo abierto limitado
Estructura	Blockchain	Tangle
Moneda	ETH	SMR
ChainId	11155111	1073
RPC URL	https://sepolia.infura.io/v3/	https://json-rpc.evm.testnet.shimmer.network
Explorador de bloques	https://sepolia.etherscan.io	https://explorer.evm.testnet.shimmer.network
Proveedor de recursos	https://sepoliafaucet.com/	https://evm-toolkit.evm.testnet.shimmer.network/

2.6.2. Contratos inteligentes

Los contratos inteligentes determinados por el diseño del prototipo fueron implementados en el lenguaje de programación Solidity en la versión 0.8.18. Se codificaron con los nombres de EnergyTransact.sol (el mercado) y Microgrid.sol (la microrred) respectivamente. Esta codificación puede ser consultada en el ***Anexo 1***: Codificación de los Contratos Inteligentes***.*** El despliegue y prueba de los contratos fue realizado utilizando la máquina virtual de Ethereum en varias redes:

1. ***Remix – Ethereum***: es el entorno de desarrollo oficial de Ethereum. Posee varias herramientas que facilitan la creación de los contratos y gracias a su intuitiva forma de conexión con la red *Blockchain* de Ethereum, permite acelerar las pruebas. El despliegue de los contratos en Remix - Ethereum puede verse en la **Figura 2-7**.
2. ***Hardhat***: Esta es una herramienta que permite la creación de una red *Blockchain* local, genera de forma automática los archivos que se deben utilizar para la implementación de la aplicación descentralizada y permite el despliegue y pruebas locales automáticas de los contratos.
3. **Sepolia Test Network,** esta es una red de pruebas oficial de Ethereum, que permite el despliegue de los contratos en un entorno seguro.
4. ***Shimmer Test Network,*** esta es la red de pruebas oficial de *IoTA*, permite el despliegue de los contratos inteligentes en un entorno seguro.

Figura 2-6 Diagrama arquitectónico de la codificación del prototipo



Elaboración propia

Las direcciones con las cuales los contratos fueron desplegados en cada una de las redes de prueba seleccionadas, se pueden ver en la **Tabla 2-4**.

Tabla 2-4 Direcciones de despliegue de contratos en las redes de prueba

Contrato Inteligente	Dirección Sepolia	Dirección Shimmer
EnergyTransact.sol	0x8C5a348af3EfEcA81d4fdc76d68188Cb97136D1B	0x74E469B69b1803a4C2e7c27A15c0c2D2f45C221b
Microgrid.sol	0x3039F0C669296359A90aFcc54c2b82EFc5190417	0x30622aD85f0126508a1459db7c6A56F33514b61B

2.6.1. Billetera digital

Una vez desplegado el contrato, se puede interactuar con el mismo, a través de una conexión en protocolo JSON-RPC, para el envío de transacciones las cuales deben ser firmadas digitalmente. La conexión a la red y la gestión de las claves requeridas para las firmas digitales son facilitadas por la *billetera digital*.

Figura 2-7 Despliegue de los contratos en Remix-Ethereum

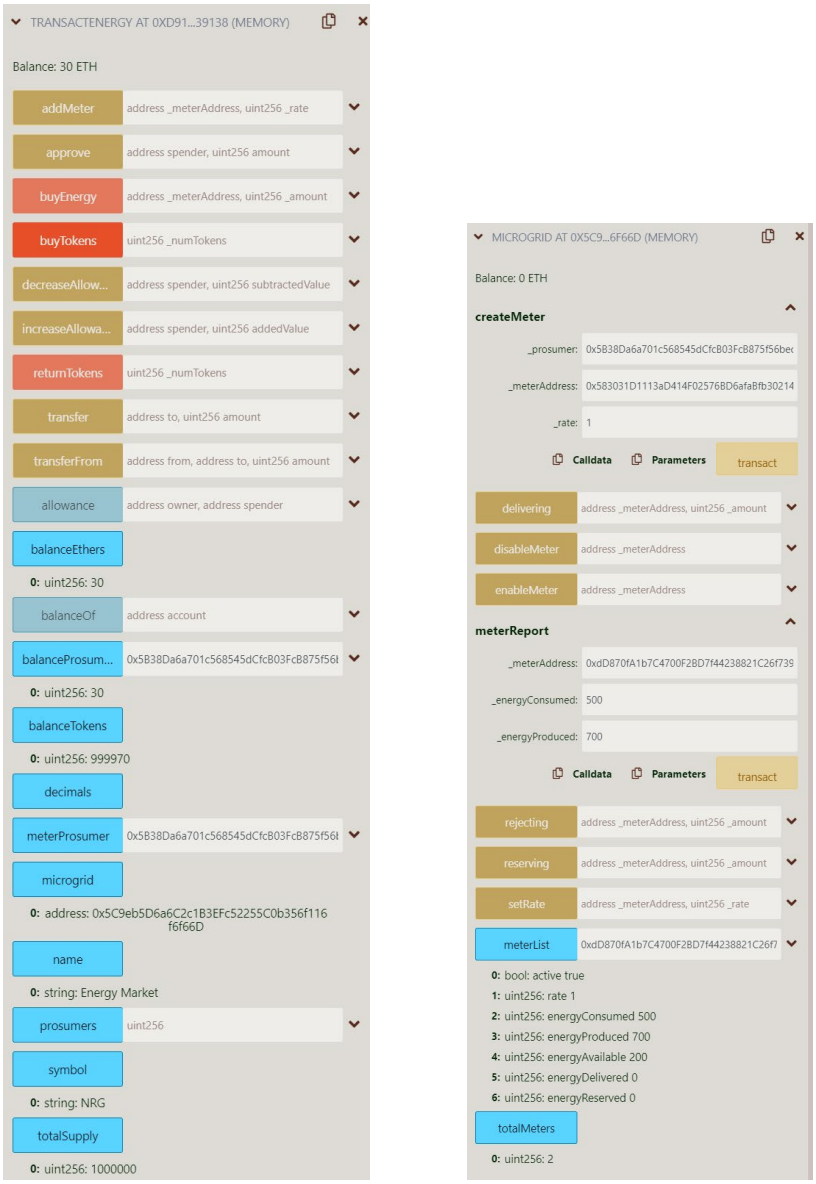


Imagen generada en Remix

Dado que se tienen dos aplicaciones que se conectan con la TRD, cada una de ellas en un ambiente distinto, se realizaron las siguientes implementaciones:

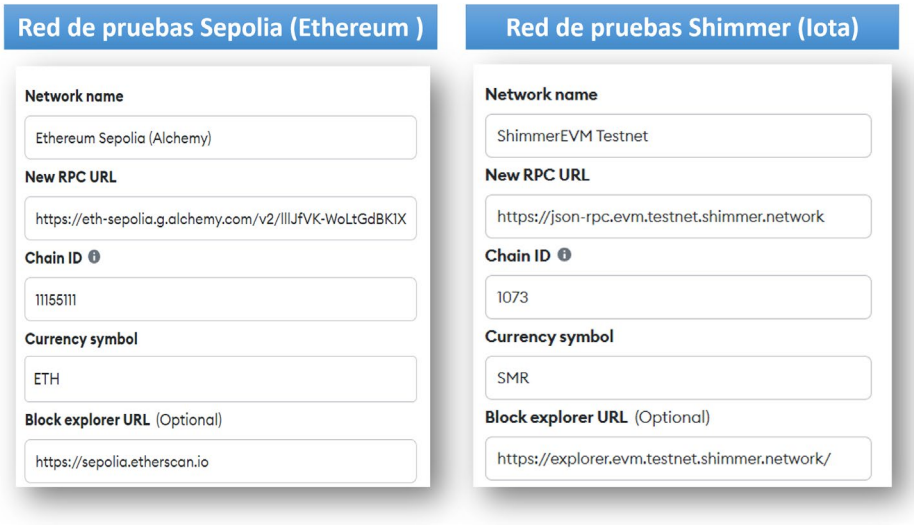
2.6.1.1. Billetera digital para la aplicación descentralizada

De acuerdo con el tipo de aplicación que se requiera implementar se puede hacer uso de una billetera digital distinta. La billetera más ampliamente utilizada para aplicaciones que

se ejecutan en un navegador es *Metamask.* Esta billetera permite a los usuarios la gestión segura de sus cuentas y claves privadas y el acceso a distintas plataformas de criptomonedas mediante una interfaz amigable e intuitiva.

Para la codificación de la aplicación descentralizada del prototipo de este proyecto se determinó el uso de la extensión para *Chrome Metamask Versión 11.7.3* como billetera digital para el acceso a las redes P2P de la aplicación descentralizada, puesto que su implementación es mediante una interfaz web que se ejecuta sobre un navegador. Dado que cada red probada tiene distintas configuraciones de acceso, se debieron ajustar las dos redes de prueba en Metamask, como se puede ver en la **Figura 2-8**.

Figura 2-8 Configuración de Metamask para acceder a las redes de prueba



2.6.1.2. Billetera digital para el Gateway

El Gateway desarrollado para la integración y/o simulación de medidores inteligentes se ejecuta en un ambiente de línea de comando, por lo tanto, se necesita un patrón de código distinto para el despliegue de la billetera digital. La billetera es desplegada mediante una función cuyos parámetros son la dirección *URL RPC* de la respectiva red y la clave privada del firmante de las transacciones. El patrón de código usado es mostrado en la **Figura 2-9**.

Figura 2-9 Patrón de código para la configuración manual de una billetera digital

```
const ethers = require("ethers");

const createContract = async (
  name,
  providerUrl,
  contractAddress,
  contractAbi,
  signerpk
) => {
  const provider = new ethers.providers.JsonRpcProvider(providerUrl);
  const signer = new ethers.Wallet(signerpk, provider);
  const contract = new ethers.Contract(contractAddress, contractAbi, signer);
  console.log(`Contrato ${name}: ${contract.address} desplegado`);
  return contract;
};
```

Pasar a la función como parámetro la ***clave privada*** de la cuenta del medidor es un aspecto que representa una vulnerabilidad para el sistema: *si la clave cae en manos de un atacante, éste podría realizar inyecciones de datos simulando medidores para que otros le compren energía, (que no va a ser entregada en la capa de campo por no estar disponible) o podría alterar el valor de energía disponible mediante los el cambio del valor de energía consumida, (con lo cual el prosumidor no podría vender sus excedentes de energía)*.

Los mecanismos posibles para el paso de la clave privada como parámetro dinámico, cada vez que se ejecute el Gateway con un medidor distinto podrían ser:

1. Almacenar las claves privadas en un archivo de ambiente (.env) que está al alcance del implementador. El riesgo es que el archivo termine en manos de un atacante (bien sea desde la máquina del implementador del sistema o bien sea por un ataque al dispositivo en el que se ejecuta el Gateway).
2. Incorporar las claves privadas como un parámetro dinámico entre los argumentos de la línea de comando al ejecutar la aplicación Gateway. Es importante tener en cuenta que las claves pueden quedar registradas en el historial de comandos del sistema operativo, al cual un atacante podría acceder.

Por considerarse menos riesgosa, se seleccionó opción 2, considerando que el acceso a la Raspberry PI, PC o servidor en donde se implemente el Gateway, requiere de credenciales de acceso, lo que reduce el riesgo de un ataque.

2.6.2. Gateway

El Gateway, se implementa como una aplicación de línea de comandos, considerando los siguientes aspectos:

- Los nodos que se conectan directamente a los medidores inteligentes generalmente son equipos compactos con conexiones a internet, pero cuyos recursos son limitados y no disponen de una pantalla para el despliegue mediante interfaz gráfica.
- El Gateway ha sido desarrollado para brindar soporte durante la etapa de evaluación como la aplicación encargada de escalar las pruebas de rendimiento a través del uso del simulador.

2.6.2.1. Codificación del Gateway

La codificación del Gateway fue realizada en *Nodejs* Versión 18.18.0 aprovechando las librerías *JSModbus* e *Ethers*.js. Mediante la librería Ethers.js se establece conexión y ejecución de transacciones sobre las instancias del contrato *Microgrid.sol* en las redes *Sepolia* y *Shimmer* respectivamente. La codificación del Gateway puede ser consultada en el repositorio descrito en **Anexo 2.**

Se implementaron dos módulos adicionales al Gateway para complementar su funcionalidad y facilitar la etapa de evaluación:

- **Simulador**: Es un módulo codificado en Nodejs responsable de la simulación valores de energía consumida y/o energía producida de acuerdo con la selección del tipo de usuario que se haga en el momento de la ejecución de la aplicación: consumidor, productor, prosumidor.
- **Registrador**: Es un módulo codificado en Nodejs responsable de reportar a una base de datos MySQL los datos de simulación: Dirección del contrato, dirección del medidor, red TRD usada, fecha y hora de envío de la transacción, estampa de tiempo de inicio de la transacción en milisegundos, estampa de tiempo de finalización de la transacción en milisegundos y el valor de latencia calculado.

2.6.2.2. Despliegue del Gateway

El despliegue de la aplicación es realizado mediante la línea de comandos del sistema operativo, para lo cual se dejaron disponibles algunos argumentos que permiten ajustar la forma como se usa la aplicación. Ellos se muestran en la **Tabla 2-5**.

Tabla 2-5 Parámetros de comandos de línea para el despliegue del Gateway

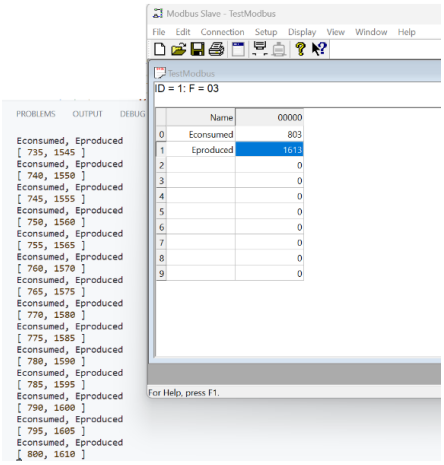
Opción	Descripción
simulated	Se usará el Gateway en modo simulación
real	El Gateway conectará un medidor inteligente real
serial	El medidor inteligente que se conectará al Gateway lo hará mediante un puerto serial
tcp	El medidor inteligente que se conectará al Gateway lo hará mediante un puerto TCP/IP
-b	Velocidad de comunicación en baudios para una conexión serial con el medidor inteligente. Ejemplo -b 9600
-s	Puerto serial de comunicación del Gateway que será usado para conectarse con el medidor inteligente. Ejemplo -s='dev/ttyUSB0'
-m	Dirección MODBUS del medidor inteligente. Ejemplo -m=1
-i	Dirección IP del medidor inteligente. Ejemplo -i=”192.168.1.12”
-p	Puerto (socket) para comunicación mediante MODBUS TCP/IP. Ejemplo -p=502
-r	El simulador se ejecutará como un medidor de un productor de energía, así que solamente se simularán valores para energía producida.
-c	El simulador se ejecutará como un medidor de un consumidor de energía, así que solamente se simularán valores para energía consumida.

2.6.2.3. Pruebas del Gateway con un simulador de MODBUS

Las primeras pruebas para validar el correcto funcionamiento del Gateway virtual fueron realizadas por medio de la aplicación Modbus Slave *V.8.2.1, Build 1954* desarrollada por la empresa *Witte Software*.

Mediante este simulador se enviaron mediante la función 03 (*holding registers*) del protocolo Modbus las dos variables que se están supervisando: Energía consumida (se usó la dirección 01) y Energía producida (se usó la dirección 02). Se implementó un mecanismo auto incremental con periodo de 10 segundos los cuales se reportaron correctamente como se puede observar en la **Figura 2-10.**

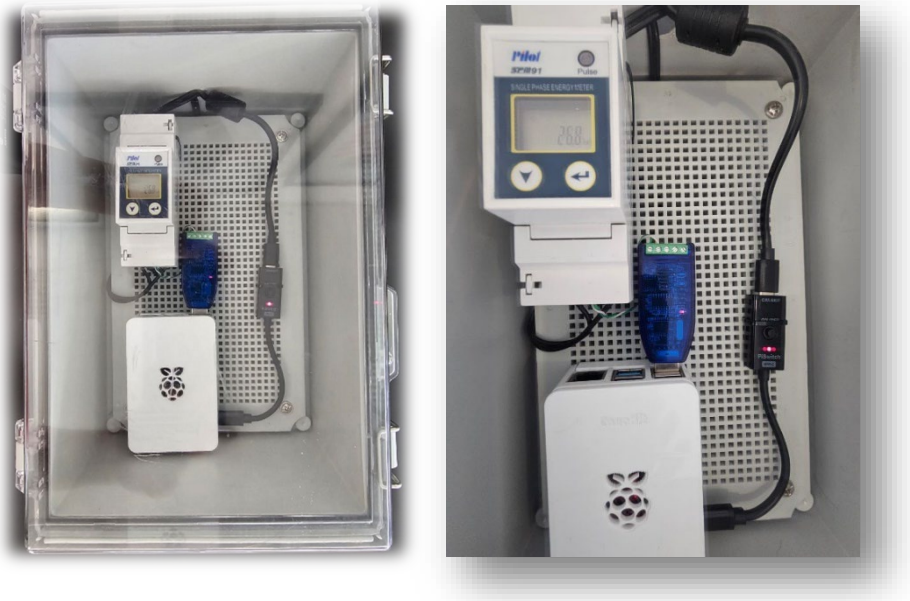
Figura 2-10 Pruebas con simulador Modbus



2.6.2.4. Pruebas del Gateway con un medidor de energía

La integración de un medidor real es realizada empleando una Raspberry Pi que ejecuta la aplicación Gateway. El medidor utilizado es un equipo ***Pilot SPM91,*** que en realidad es unidireccional por lo que se ajustó el código para que reportara únicamente la energía consumida. La conexión física de los dispositivos se hizo mediante un enlace RS485 conectado al puerto serial de la Raspberry mediante un convertidor USB a RS485 como se ve en la imagen de la . El protocolo de comunicaciones utilizado es MODBUS RTU.

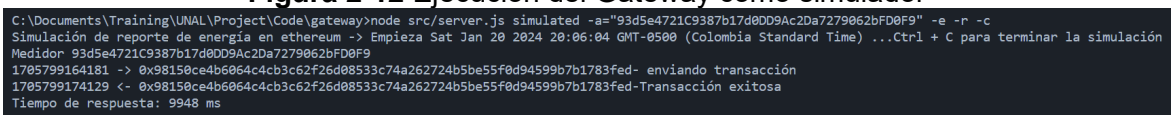
Figura 2-11 Implementación de comunicación serial con un medidor



2.6.2.5. Pruebas del Gateway como simulador

La aplicación Gateway incluyó un módulo, el cual implementó un simulador de medidor de energía bidireccional. Con ello, se pudieron adelantar las pruebas sin incurrir en gastos y tiempo de configuración y cableado de medidores reales. En la **Figura 2-12** es posible ver el despliegue del Gateway durante una prueba.

Figura 2-12 Ejecución del Gateway como simulador



2.6.3. Aplicación descentralizada

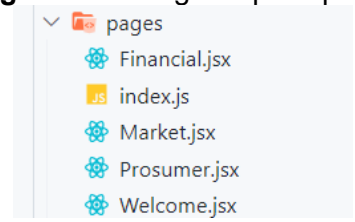
A pesar de que se desarrolló una interfaz inicial única para el sistema de energía transactiva, para el despliegue final se particularizó cada aplicación de acuerdo con los parámetros del mercado de cada TRD y se ajustó el aspecto para facilitar al usuario la identificación de la TRD en prueba.

2.6.3.1. Codificación de la aplicación descentralizada

La aplicación descentralizada encargada de la gestión de la información del mercado de energía y de las transacciones de intercambio entre prosumidores es una aplicación Web que puede ser renderizada en la mayoría de los navegadores modernos: Chrome, Firefox, Edge, Safari, etc. de acuerdo con (*JavaScript Environment Requirements – React*, s/f). Fue codificada usando React V.18.2.0, Vitejs V.4.2.0, Ethers.js V.5.7.2. y tiene como prerrequisito para su codificación y despliegue el uso de navegadores compatibles con la billetera digital seleccionada *Metamask:* Chrome, Firefox, Brave, Edge, Opera.

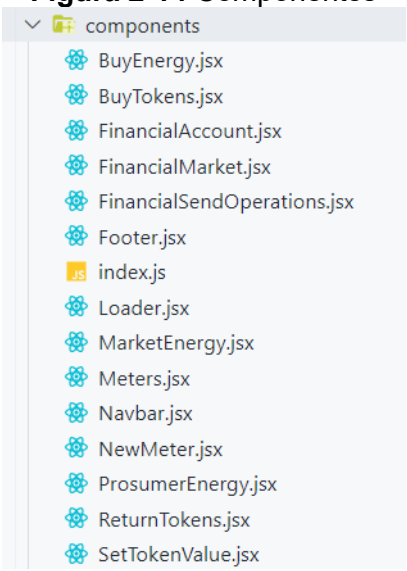
De acuerdo con la descomposición funcional del diseño (mostrada en la **Figura 2-5**) se desarrollaron los módulos que estructuran la navegación principal. Ellos son mostrados en la **Figura 2-13**.

Figura 2-13 Páginas principales



De acuerdo con los patrones de diseño y codificación de *React* los módulos principales, se estructuran a partir de componentes individuales, cada uno de los cuales gestiona justo la información requerida, lo que agiliza el renderizado de la aplicación en el explorador del cliente. Los componentes implementados se muestran en la **Figura 2-14**.

Figura 2-14 Componentes



Finalmente, todos los procesos asociados a la conexión y gestión de transacciones en la TRD se codificaron en un contexto único: *DltContext.jsx*

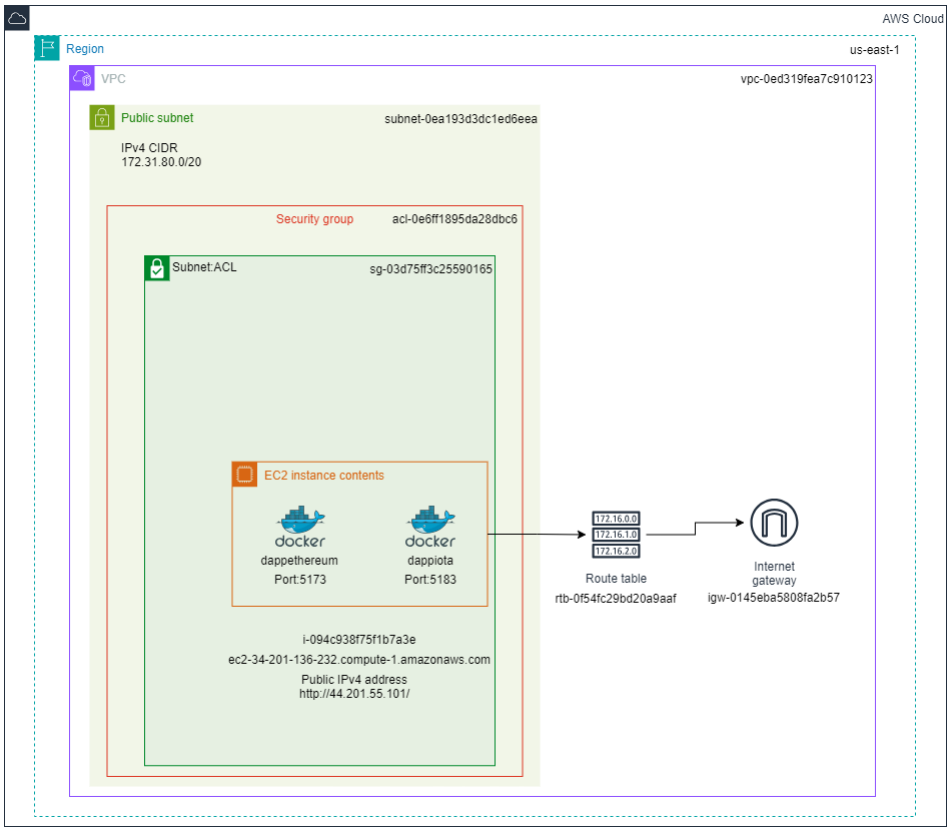
La codificación de las versiones de aplicación descentralizada puede ser consultada en los repositorios descritos en el **Anexo 2**

2.6.3.2. Despliegue de la aplicación descentralizada

Cada aplicación descentralizada codificada fue empaquetada en un contenedor Docker, que facilita su virtualización y publicación en la nube.

Para efectos de probar el prototipo en una nube real, y haciendo uso de la IaaS (Infraestructura como servicio) ofrecida por AWS se implementó una infraestructura mínima que consta de una nube de cómputo elástica básica (EC2) la cual se comunica con internet a través de un enrutador virtual y un internet Gateway. La EC2 cuenta con una dirección IP pública que permite el acceso de cualquier usuario a través de internet desde cualquier lugar del mundo. La representación de la implementación realizada puede verse en la **Figura 2-15**.

Figura 2-15 Despliegue de la aplicación descentralizada en la nube AWS



El prototipo de aplicación descentralizada se renderizó en dos versiones para efectos de facilitar la identificación por el usuario durante el proceso de evaluación. en las figuras **Figura 2-16** y **Figura 2-17** se pueden observar las capturas de pantalla de las páginas principales de cada aplicación descentralizada.

En el **Anexo 3** se realiza una descripción detallada del uso de la interfaz, la cual permitirá conocer los distintos despliegues disponibles para cumplir con la funcionalidad del prototipo.

Figura 2-16 Renderizado de la aplicación descentralizada para pruebas con Ethereum

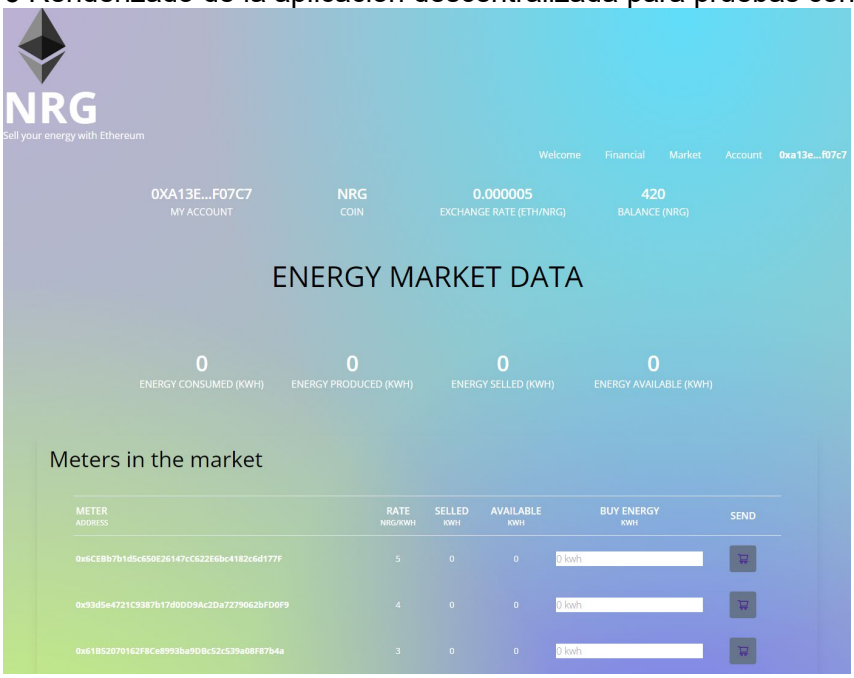
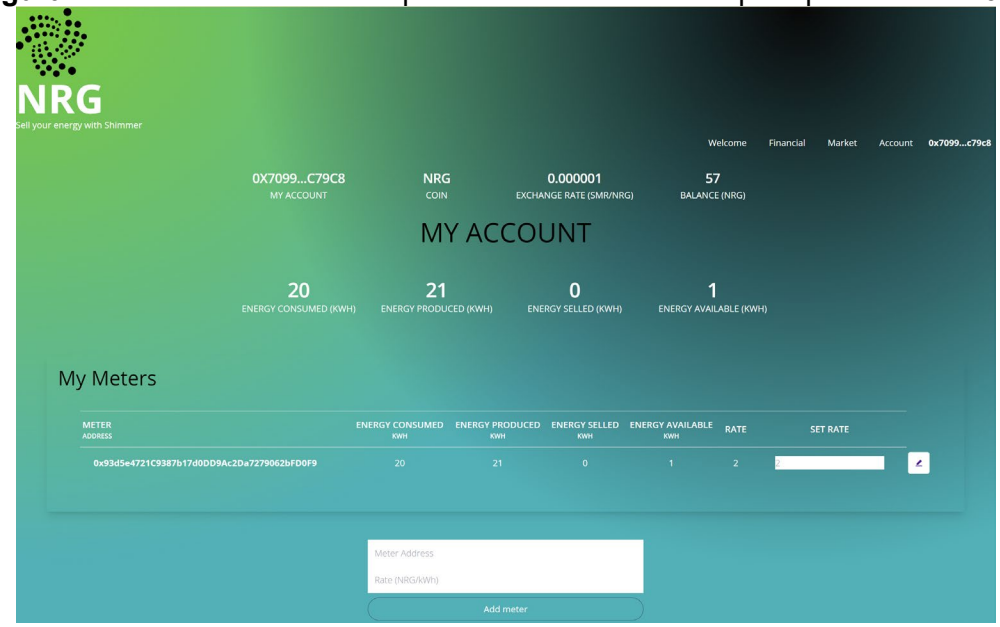


Figura 2-17 Renderizado de la aplicación descentralizada para pruebas con *IoTA*



Este capítulo consolida los aspectos considerados para el diseño, implementación y pruebas del prototipo de sistema de energía transactiva distribuida. Se ha documentado la implementación del sistema de energía transactiva en dos TRD que presentan características de madurez, estabilidad y escalabilidad distintas. La selección de las TRD se basó en su capacidad para utilizar contratos inteligentes y aplicaciones descentralizadas, lo que permitió reducir el espacio de búsqueda entre las TRD existentes. Finalmente, se decidió seleccionar Ethereum debido a la gran cantidad de aplicaciones que la utilizan con éxito, mientras que la elección de IoTA representa una oportunidad para validar una TRD emergente que incluyó recientemente la posibilidad de utilizar contratos inteligentes aprovechando la MVE. Es indispensable evaluar el comportamiento de estas dos versiones del prototipo, para ello en el siguiente capítulo se identificará la metodología para hacerlo, se ejecutarán las pruebas y se documentarán los resultados.

3. Evaluación y pruebas

El tercer y último objetivo de este trabajo es evaluar el desempeño del prototipo de sistema de energía transactiva en un ambiente simulado. Esta evaluación tiene un propósito adicional: comparar el desempeño entre las dos TRD seleccionadas: Ethereum e *IoTA*. Se pretende probar que el prototipo de sistema de energía transactiva puede ser implementado y ejecutado tanto en una TRD madura como Ethereum como en una TRD emergente.

3.1. Preparación del escenario de pruebas

El proceso de evaluación requiere una preparación preliminar del escenario de las pruebas de forma que se habilite al sistema para la ejecución de las simulaciones, se garantice que los datos guardan coherencia y por consiguiente, se evite que las transacciones sean rechazadas por los contratos inteligentes en tiempo de ejecución.

3.1.1. Usuarios y equipos

Los recursos iniciales que permiten la simulación de las pruebas en cada TRD, incluyen la implementación de una pequeña comunidad de diez prosumidores cada uno con un medidor (ver la en la **Tabla 3-1**).

Tabla 3-1 Tabla de prosumidores y medidores para pruebas

Prosumidor	Dirección de la cuenta	Medidor	Medidor
Account01	0xc2A15a7f6cD89E22286720976eb699280bEA2dEA	Met01	0x6CEBb7b1d5c650E26147cC622E6bc4182c6d177F
Account02	0xA13e5EAf1d0D08C3f79a6033351Ba47401Cf07C7	Met02	0x93d5e4721C9387b17d0DD9Ac2Da7279062bFD0F9
Account03	0x1Cf16A18C028937e516f7bB380B2EE4C73339976	Met03	0x61B52070162F8Ce8993ba9DBc52c539a08F87b4a
Account04	0x6B4B1D3E88381526C9593ea4b0D2bb98836934C4	Met04	0xD8FEE45D700A4BDe086189846889cDeD99d66508
Account05	0xE5478a80EFa3a08038eBD6674D319aE5138ADf42	Met05	0xB6C034e04Ff43acAC14d5cfdBbeE7A580d2aF2D9
Account06	0x828D07144B65c8E77C101ED41Ae637DcFE69c703	Met06	0x9aaA94e06cAe4eC8BA2F67dA68b7AF70372AC833
Account07	0x22d7dC1c7905ef72400c278e7fc5d9212C369F94	Met07	0xDcabCb2A5Bbd2b61616cf7916904af0D03852C1A
Account08	0xFF101916ffF0b5fDEeF0150981b0BbDE90146643	Met08	0x8526740A4B8d2c4078a0F8cDeDf5beaeDa2c3975
Account09	0xcbBf6f3E2500C50979Ba032AE5bAEC5b1C72F41D	Met09	0x1Ab633070cE458233D8eCdc0B846d183d4F3011e
Account10	0xDa6cB67FF454AB2756b310402EE4f31700aC0778	Met10	0xd157cE331685DE22D751fdac1Ccf87A9D4c45630

Elaboración propia

Para ello se crean las cuentas y se obtiene la dirección y clave privada de cada una de ellas. A continuación, se deben conseguir para cada cuenta los recursos en ETH y SMR que solventen los pagos de las transacciones en cada TRD durante el período de pruebas. Estos recursos son obtenidos de los respectivos proveedores de recursos listados en la **Tabla 2-3** Comparativo de características de las TRD seleccionadas.

3.1.2. Preparación del mercado

Mediante la aplicación descentralizada, se ejecutaron las siguientes operaciones en cada una de las redes de prueba que permitieron la preparación del mercado y la prueba de la correcta ejecución de las operaciones:

- Creación de medidores con las direcciones habilitadas en la tabla **Tabla 3-1**.
- Compra de 1000 tokens NRG por cuenta de cada prosumidor para comercialización de energía
- Devolución de 100 tokens a la red de prueba

Las transacciones derivadas de este ejercicio pueden ser consultadas en el **Anexo 4**

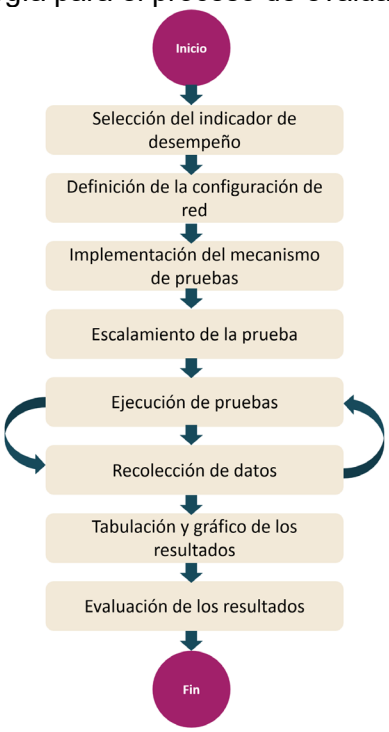
3.2. Evaluación de desempeño

La metodología usada para la evaluación de los prototipos metodología es mostrada en la **Figura 3-1** y cuyos pasos son descritos a continuación:

3.2.1. Selección del indicador de desempeño

Con el fin de realizar la evaluación de desempeño del prototipo de sistema de energía transactiva se definió que se utilizaría como indicador *la latencia de transacciones de escritura*. Una transacción de escritura es una transición de estado en donde mediante una función algunos datos en la TRD adquieren un nuevo valor. Las transacciones son enviadas por el sistema de energía transactiva y son validadas y aprobadas por la TRD mediante el algoritmo de consenso.

Figura 3-1 Metodología para el proceso de evaluación de desempeño



Elaboración propia

La latencia de transacción o de escritura L_t es definida como la diferencia entre el tiempo de confirmación de la transacción T_t y el tiempo de envío de la solicitud T_s (Abdella et al., 2021).

$$L_t = T_t - T_s \qquad \textit{(Ecuación 3-1)}$$

Es importante evaluar la latencia de las transacciones puesto que permite calcular el desempeño de la aplicación considerando los tiempos de propagación y convergencia del algoritmo de consenso de la TRD seleccionada.

3.2.2. Definición de la configuración de red

En cuanto a la configuración de la red se pueden tener tres escenarios:

1. *Usando una Blockchain local*: este escenario tiene como beneficio que es muy fácil de implementar y no acarrea gastos adicionales puesto que se hace en un ambiente simulado localmente en un computador. Sin embargo, las pruebas efectuadas en este escenario quedan limitadas a la capacidad de cómputo del equipo en el que se ejecutan.

2. *Usando varios nodos existentes en las redes de prueba de la TRD:* Este escenario presenta la ventaja de ser fácil de configurar, ya que no requiere la preparación de nodos. Además, las pruebas se aproximan más a una ejecución real, pero en un entorno seguro.
3. *Implementando varios nodos cada uno de ellos ejecutado en distinta instancia de nube.* Este escenario implica la reproducción de la configuración de nube por nodo, así como la implementación de una red coherente y ajustes más complejos de seguridad y control de acceso para asegurar la comunicación entre los nodos. Además, conlleva costos económicos más elevados debido al requerimiento de mayores recursos en la nube.

Se decidió la opción de las redes de prueba de cada TRD lo cual facilita y enfoca los esfuerzos del proceso de evaluación en la prueba del sistema de energía transactiva y no en la preparación de las redes P2P que soporten las TRD seleccionadas.

3.2.3. Implementación del mecanismo de pruebas

Evaluar una aplicación descentralizada basada en TRD es un proceso complejo debido a que estas tecnologías son relativamente nuevas y abarcan una amplia gama de aspectos a considerar (Fan et al., 2020). Estos aspectos incluyen la red peer-to-peer (P2P), los nodos utilizados como infraestructura, los algoritmos de consenso, los tipos de transacciones a probar y la propia implementación de las aplicaciones y contratos inteligentes. Ya existen estudios y plataformas que han llevado a cabo comparaciones entre TRD utilizando diversas métricas, tales como la tasa de éxito, la latencia, la tasa de transferencia efectiva, el consumo de recursos, y la cantidad de transacciones por segundo que le permiten al desarrollador de aplicaciones elegir la TRD apropiada para la implementación de sus aplicaciones. En la **Tabla 3-2** son listadas algunas de ellas. Sin embargo, estas evaluaciones comparativas generalmente obtienen sus resultados en un entorno aislado, donde las pruebas no incluyen el rendimiento del contrato inteligente o la aplicación distribuida en su totalidad. Es decir, la prueba termina siendo una evaluación de la TRD en sí misma. La ausencia de una documentación detallada y de un mantenimiento activo en estas plataformas, así como su alcance limitado a TRD, impidieron su utilización para la evaluación del prototipo de energía transactiva.

Tabla 3-2 Herramientas de evaluación de desempeño

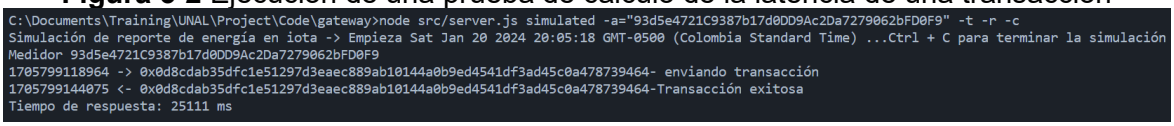
	Métricas disponibles								Compatibilidad con TRD														
Herramientas de evaluación	Tasa de éxito	Throughput	Latencia	Uso de recursos	Bien documentada	Mantenimiento activo	Cargas de trabajo Fijas	Cargas de trabajo personalizables	Algorand	Avalanche	BitShares	Diem	Ethereum	FISCO BCOS	Hyperledger	Parity	Quorum	RedBelly	Solana	Stellar	IoTA	Nano	Byteball
BCTMark	X	X	X	X																			
Blockbench	X	X	X	X	X		X						X		X	X	X						
Chainhammer		X											X			X	X						
DAGBench		X	X	X																	X	X	X
DIABLO		X	X					X	X	X		X	X		X		X	X	X				
Gromit		X	X	X					X	X	X	X	X		X					X			
HyperLedger Caliper					X	X	X	X					X	X	X		X						

Elaboración propia basada en datos obtenidos de (Fan et al., 2020), (Abdella et al., 2021) y (Pedro & Lopes, 2023).

De acuerdo con lo anterior, se toma la decisión de desarrollar un mecanismo personalizado para este prototipo, cual permite el cálculo de la latencia de transacciones de escritura sobre el contrato inteligente *Microgrid.sol.* Para lograrlo, se utiliza la aplicación Gateway en modo simulación, y se configura para que envíe transacciones automáticamente cada 5 minutos a la respectiva TRD por cada medidor en la microrred de la comunidad, según lo especificado en la **Tabla 3-1** Tabla de prosumidores y medidores para pruebas.

Para medir la latencia de esta simulación, la aplicación Gateway registra la marca de tiempo de envío de la transacción T_s y la marca de tiempo de finalización de la transacción T_t (cuando es añadida a la estructura de datos de la TRD), a partir de las cuales se calcula la latencia utilizando la (*Ecuación 3-1)*. En la **Figura 3-2** se puede ver una de las pruebas de cálculo de latencia realizadas.

Figura 3-2 Ejecución de una prueba de cálculo de la latencia de una transacción



3.2.4. Registro de la información de las pruebas

Con el objetivo de registrar cada una de las pruebas en una única plataforma que facilite la posterior tabulación de la información, se llevaron a cabo las siguientes implementaciones:

1. Se configuró una instancia de base de datos relacional MySQL en AWS RDS. En esta instancia se desplegó una base de datos denominada "evaluationdb", la cual contiene una tabla llamada "transactions". Esta tabla se encarga de almacenar los registros reportados por la aplicación Gateway durante las simulaciones
2. Se adicionó al código del Gateway un mecanismo de conexión y creación de registros de transacciones a la base de datos MySQL
3. Se modificó la implementación del Gateway para que pueda recibir como argumentos en el código de línea de la simulación la clave privada del medidor *(SIGNERPK) y* la clave de la base de datos *(PWDB).* Una de estas pruebas de ejecución se puede observar en la **Figura 3-3**.

Figura 3-3 Ejemplo de comandos de línea del Gateway

```
ubuntu@ip-172-31-33-195:~$ sudo docker run -e METERADDRESS="9aaA94e06cAe4eC8BA2F67dA68b7AF70372AC833" -e SIGNERPK="af78f011063
e27f14f0cea0b0e80[redacted]" -e PWDB="[redacted]" -e NETWORK="-e" --name ethmet06 1324bf37da97
Contrato 0x3039F0C669296359A90aFcc54c2b82EFc5190417
Medidor 9aaA94e06cAe4eC8BA2F67dA68b7AF70372AC833
Simulación de reporte de energía en ethereum -> Empieza Mon Jan 22 2024 04:12:28 GMT+0000 (Coordinated Universal Time) ...Ctrl
 + C para terminar la simulación

Connection established
```

4. Se construyó una imagen base de Docker, la cual fue ejecutada mediante parámetros dinámicos pasados como los argumentos durante el despliegue del contenedor. De esta manera, cada contenedor ejecutado con parámetros específicos simula a un medidor en una TRD. Finalmente, en la **Figura 3-4** se muestra una captura de pantalla de la ejecución de todos los contenedores en la máquina virtual de AWS donde se desplegó la prueba. Se puede observar que cada uno de los contenedores fue nombrado de acuerdo con el medidor que simulan y la red de pruebas en la que se está ejecutando. Así mismo, en la imagen se evidencia el identificador de la imagen de Docker que sirve como base: "*1324bf37da97".*

Figura 3-4 Modelo de integración Docker del módulo Gateway para simulación

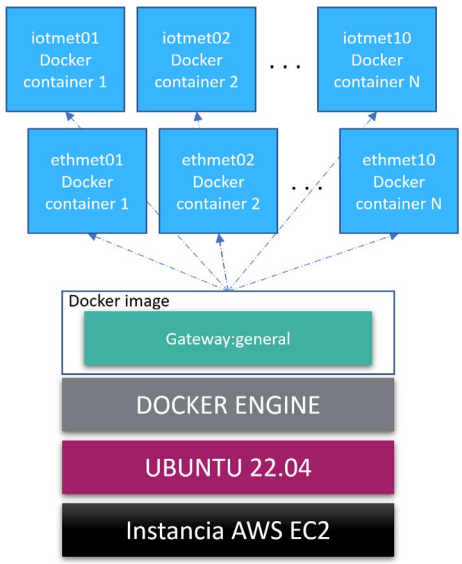


Figura 3-5 Despliegue real los contenedores para pruebas

```
ubuntu@ip-172-31-33-195:~$ sudo docker container ps
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS       PORTS     NAMES
b363b9e33caa   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             iotamet10
7b9fd8f36a47   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             ethmet10
7dc1c80009b7   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             iotamet09
542e0f4c91aa   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             ethmet09
51d915d80057   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             iotamet08
a8f3a363e570   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             ethmet08
aacaaf3c838b   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             iotamet07
1d6c9394323e   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             ethmet07
9b57d94ba332   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             iotamet06
36b744c2c0b0   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             ethmet06
ecf4402729fc   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             iotamet05
5d61e945a29d   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             ethmet05
69ada0f6d112   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             iotamet04
e9064cd7837b   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             ethmet04
0eed60099d37   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             iotamet03
c871476ab8e3   1324bf37da97   "docker-entrypoint.s…"   6 hours ago   Up 6 hours             ethmet03
c92a69462a26   1324bf37da97   "docker-entrypoint.s…"   7 hours ago   Up 7 hours             iotamet02
22e66532db68   1324bf37da97   "docker-entrypoint.s…"   7 hours ago   Up 7 hours             ethmet02
863dec6aa481   1324bf37da97   "docker-entrypoint.s…"   7 hours ago   Up 7 hours             iontamet01
abbfcd51f7e8   1324bf37da97   "docker-entrypoint.s…"   7 hours ago   Up 7 hours             met01
```

3.2.5. Ejecución de las pruebas

La ejecución de las pruebas fue realizada durante un período de 5h con una cantidad de 10 medidores reportando valores de energía cada 5 minutos. Es importante resaltar que las pruebas fueron realizadas en un horario de bajo tráfico, para poder optimizar los recursos que cada una de las cuentas de los medidores. Sin embargo, las pruebas están limitadas por el límite de gas configurado y por los recursos disponibles en cada cuenta para el pago de las expensas por aprobación de transacciones que requiere la TRD.

3.2.6. Recolección de datos

Los datos de cada transacción fueron reportados de forma automática por la aplicación Gateway de simulación hacia la base de datos de AWS RDS MySQL en donde quedaron almacenados y disponibles para ser descargados para su análisis.

La información recolectada incluye la estampa de tiempo (en UTC), el hash de la transacción efectuada, los valores de energía consumida y producida, la estampa de tiempo de solicitud de la operación, la estampa de tiempo de confirmación de la operación y la latencia calculada para cada registro. La información pudo ser validada y descargada mediante la aplicación *Oracle MySQL Workbench*, como se puede ver en la **Figura 3-6**.

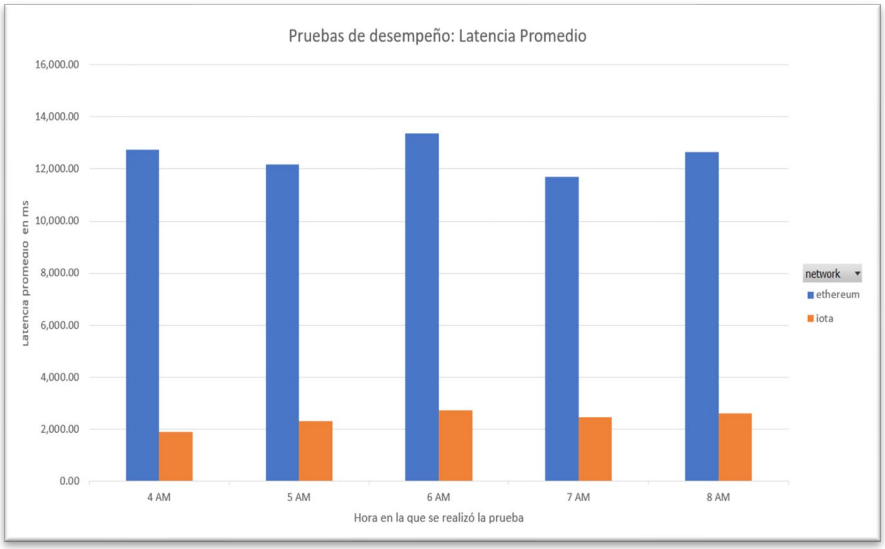
Figura 3-6 Información almacenada en la base de datos *evaluationdb* – MySQL

contract	meteraddress	datetimestamp	transactionhash	econsumed	eproduced	initialtimestamp	finaltimestamp	latency
0x30622aD85f0126508a1459db7c6A56F33514b61B	6CEBb7b1d5c650E26147cC622E6bc4182c6d177F	2024-01-21 21:52:05	0x41b7b69afdb7a67cdd89af05cb93d883c06f0088287	1	8	1705891919822	1705891925402	5580
0x3039F0C669296359A90aFcc54c2b82EFc5190417	6CEBb7b1d5c650E26147cC622E6bc4182c6d177F	2024-01-22 03:59:50	0xf8d0dbb804b204e837a0fcdd9204c4ece645eab3509	33	72	1705895977122	1705895989830	12708
0x30622aD85f0126508a1459db7c6A56F33514b61B	6CEBb7b1d5c650E26147cC622E6bc4182c6d177F	2024-01-22 04:01:08	0xfc02c06ea1bd3d5c7c5241343d4830fa9cab0d7fcad	7	18	1705896065578	1705896067892	2314
0x3039F0C669296359A90aFcc54c2b82EFc5190417	93d5e4721C9387b17d0DD9Ac2Da7279062bFD0F9	2024-01-22 04:03:39	0xd51444798b9651a648e4c0ee39b1c83a3e06ed814d6	3	9	1705896206696	1705896219384	12688
0x30622aD85f0126508a1459db7c6A56F33514b61B	93d5e4721C9387b17d0DD9Ac2Da7279062bFD0F9	2024-01-22 04:04:18	0x5de9ff38a26ce346c5bcb5babf3f62cad9985915353	3	8	1705896255203	1705896257556	2353
0x3039F0C669296359A90aFcc54c2b82EFc5190417	6CEBb7b1d5c650E26147cC622E6bc4182c6d177F	2024-01-22 04:04:50	0x2fc9bb41d24fd9369657f3eae48c97cef9473672338	34	85	1705896277220	1705896289828	12608
0x30622aD85f0126508a1459db7c6A56F33514b61B	6CEBb7b1d5c650E26147cC622E6bc4182c6d177F	2024-01-22 04:06:07	0xf82f2c751e4abb837b546ba0196984647ac81ba64e0	14	23	1705896365427	1705896367229	1802
0x30622aD85f0126508a1459db7c6A56F33514b61B	61B52070162F8Ce8993ba9DBc52c539a08F87b4a	2024-01-22 04:08:38	0x400825b8f69cbc9e21485dc04981b1a804cae681050	4	11	1705896516129	1705896518426	2297
0x3039F0C669296359A90aFcc54c2b82EFc5190417	93d5e4721C9387b17d0DD9Ac2Da7279062bFD0F9	2024-01-22 04:08:39	0x791b13b214199abc54ced4f7e6571d95ac4896134a6	12	15	1705896506784	1705896519395	12611
0x30622aD85f0126508a1459db7c6A56F33514b61B	93d5e4721C9387b17d0DD9Ac2Da7279062bFD0F9	2024-01-22 04:09:17	0x3624a33e4ccf90be84c0809ca8c164874f79841a08b	8	15	1705896555226	1705896557065	1839
0x3039F0C669296359A90aFcc54c2b82EFc5190417	6CEBb7b1d5c650E26147cC622E6bc4182c6d177F	2024-01-22 04:09:50	0x010bf5cf647b510d555c36d4036722a4a8591946d21	42	96	1705896577347	1705896590026	12679
0x30622aD85f0126508a1459db7c6A56F33514b61B	6CEBb7b1d5c650E26147cC622E6bc4182c6d177F	2024-01-22 04:11:08	0x3c9230d811f236411d6b3f4fde73e7ea0a4852907a5	22	25	1705896665783	1705896668097	2314
0x3039F0C669296359A90aFcc54c2b82EFc5190417	D8FEE45D700A4BDe08618984689cDeD99d66508	2024-01-22 04:11:26	0x4bc70df19a2c00ba3881812263d17a003fc0c1d5bf6	6	7	1705896681427	1705896686088	4661
0x3039F0C669296359A90aFcc54c2b82EFc5190417	61B52070162F8Ce8993ba9DBc52c539a08F87b4a	2024-01-22 04:13:16	0x4449923072231f2eab8722f4eebee626b54325c62c59	0	4	1705896779393	1705896796046	16653
0x30622aD85f0126508a1459db7c6A56F33514b61B	61B52070162F8Ce8993ba9DBc52c539a08F87b4a	2024-01-22 04:13:38	0x8e36f3febb22452fbfbb573c54de823a4e1046114e3	5	20	1705896816199	1705896818250	2051
0x3039F0C669296359A90aFcc54c2b82EFc5190417	93d5e4721C9387b17d0DD9Ac2Da7279062bFD0F9	2024-01-22 04:13:39	0xb7bc031db56daab512ddeed0c80ae1d172e1afcf4cd	12	22	1705896806877	1705896819477	12600
0x3039F0C669296359A90aFcc54c2b82EFc5190417	B6C034e04Ff43acAC14d5cfd8beE7A580d2aF2D9	2024-01-22 04:14:14	0xb4b4fae27e0c004990b9bcf6d5c41ac3be0c8cd8d52	2	5	1705896848875	1705896853531	4656
0x30622aD85f0126508a1459db7c6A56F33514b61B	93d5e4721C9387b17d0DD9Ac2Da7279062bFD0F9	2024-01-22 04:14:19	0xfd83b3d126e2c4cfd1507620710b81b2d8f8a7f032a	14	20	1705896855780	1705896858562	2782
0x30622aD85f0126508a1459db7c6A56F33514b61B	B6C034e04Ff43acAC14d5cfd8beE7A580d2aF2D9	2024-01-22 04:14:31	0xe56049e9abaff524088f034132f16fe5c8a2a4b5cdb	3	3	1705896868397	1705896870972	2575
0x3039F0C669296359A90aFcc54c2b82EFc5190417	6CEBb7b1d5c650E26147cC622E6bc4182c6d177F	2024-01-22 04:14:50	0x10df49639f7c201dcd4e57127055f5a8950274b4aa8	47	107	1705896877403	1705896890041	12638
0x30622aD85f0126508a1459db7c6A56F33514b61B	6CEBb7b1d5c650E26147cC622E6bc4182c6d177F	2024-01-22 04:16:08	0x9c7fb09e436940e064c673c4eb971b0def14aaa0230	28	29	1705896966128	1705896968472	2344
0x3039F0C669296359A90aFcc54c2b82EFc5190417	9aaA94e06cAe4eC8BA2F67dA68b7AF70372AC833	2024-01-22 04:17:38	0x289a9f364d5750a4d9be36f77fd8e0281a66032664d	0	2	1705897049231	1705897057856	8625
0x30622aD85f0126508a1459db7c6A56F33514b61B	D8FEE45D700A4BDe08618984689cDeD99d66508	2024-01-22 04:17:51	0x04780a8bfe93981bee02222fd52f4c4517a5e59f789	9	9	1705897069530	1705897070839	1309
0x3039F0C669296359A90aFcc54c2b82EFc5190417	61B52070162F8Ce8993ba9DBc52c539a08F87b4a	2024-01-22 04:18:16	0x8c5af4a8810f608c4fdd481b1fadf6e7bec06b92ef2	6	12	1705897079460	1705897096120	16660
0x30622aD85f0126508a1459db7c6A56F33514b61B	61B52070162F8Ce8993ba9DBc52c539a08F87b4a	2024-01-22 04:18:39	0x2db4d9e2cdbfcdcbf6ed50ccadf8774ec7d193d0860	14	27	1705897116284	1705897118547	2263
0x3039F0C669296359A90aFcc54c2b82EFc5190417	93d5e4721C9387b17d0DD9Ac2Da7279062bFD0F9	2024-01-22 04:18:40	0xb79dae746e88452bf579776ce2d4ac22b61cbf69f73	14	35	1705897106983	1705897119671	12688
0x30622aD85f0126508a1459db7c6A56F33514b61B	9aaA94e06cAe4eC8BA2F67dA68b7AF70372AC833	2024-01-22 04:18:57	0x9382c9f01815df8177b8b9aedf973a0138cc0f49237	5	6	1705897135632	1705897137475	1843
0x3039F0C669296359A90aFcc54c2b82EFc5190417	B6C034e04Ff43acAC14d5cfd8beE7A580d2aF2D9	2024-01-22 04:19:14	0x7c592bb370a79199f9cf64ef15b66bcfce03f9ae152	10	15	1705897148933	1705897153555	4622
0x30622aD85f0126508a1459db7c6A56F33514b61B	93d5e4721C9387b17d0DD9Ac2Da7279062bFD0F9	2024-01-22 04:19:18	0xaf4f0a6d46c807bf31d2331819a3b5a6767e9a14286	14	34	1705897155399	1705897157909	2510

3.3. Tabulación y representación gráfica de los resultados

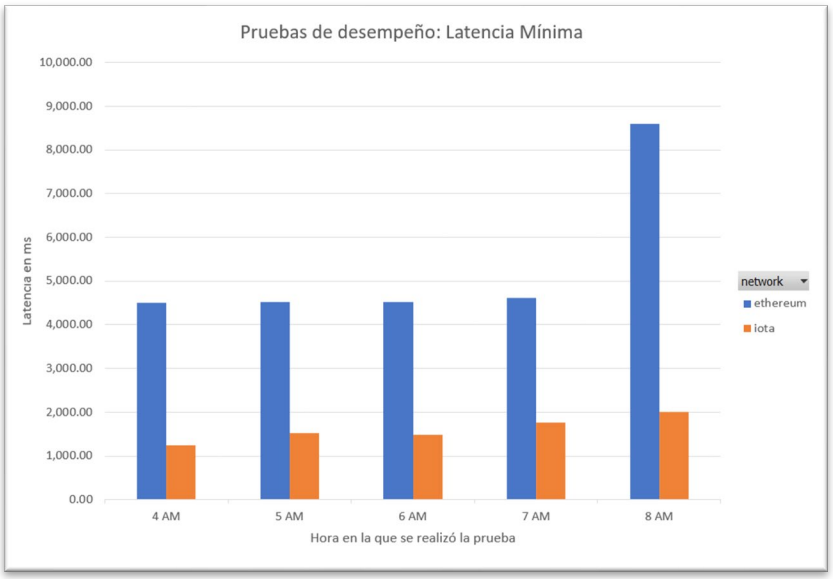
A partir de la información recopilada, se creó una tabla pivote que permitió agrupar las transacciones realizadas por hora de prueba y calcular el valor promedio, máximo y mínimo de la latencia por hora para todos los 10 simuladores de medidores implementados durante las 5 horas de la prueba. Los resultados de esta tabulación permitieron la generación de las figuras: **Figura 3-7**, **Figura 3-8** y **Figura 3-9**, las cuales son objeto de análisis.

Figura 3-7 Gráfica de Latencia promedio por hora de pruebas



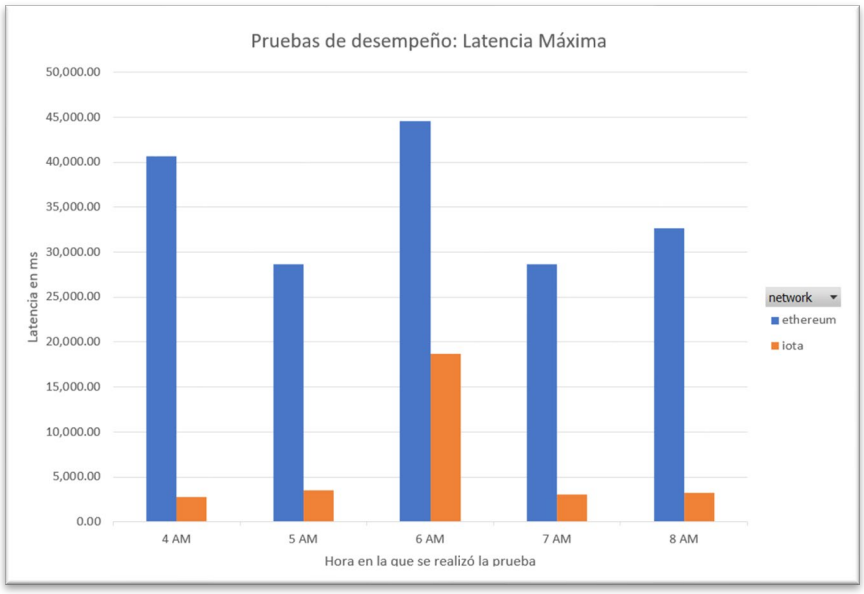
Elaboración propia

Figura 3-8 Gráfica de Latencia mínima por hora de pruebas



Elaboración propia

Figura 3-9 Gráfica de Latencia máxima por hora de pruebas



Elaboración propia

3.4. Evaluación de los resultados

La **Figura 3-7** muestra que los valores de latencia promedio en la ejecución de transacciones de escritura en cada una de las TRD evaluadas son consistentes: se

evidencia una tendencia de tiempos consistente en cada hora de prueba. Es clara una diferencia marcada en la latencia promedio por hora entre Ethereum e IoTA: una relación de 5 a 1. Las transacciones ejecutadas en Ethereum tienen una latencia promedio por hora de 12.6 segundos, mientras que aquellas que fueron ejecutadas en *IoTA* tienen 2.35 segundos.

La gráfica de latencia mínima por hora permite deducir en el mejor de los casos Ethereum puede tener una latencia de 4.5 segundos, mientras que *IoTA* podría tener 1.2 segundos.

Y finalmente, la gráfica de latencia máxima por hora permite deducir que se puede llegar a tener una latencia máxima de 45 segundos en Ethereum y una de 18.7 segundos en *IoTA*.

Se puede evidenciar un rendimiento consistentemente mejor de las transacciones de escritura realizadas con el contrato *Microgrid.sol* cuando se ejecutan en IoTA comparado con su ejecución en Ethereum.

3.5. Comparación de resultados con un estudio anterior

En el capítulo 2 de este documento, se presentó la Tabla 1-4, comparativa entre distintas TRD, en la cual se incluyen los tiempos promedio creación de un bloque que, de acuerdo con (Jabed Morshed Chowdhury et al., s/f), está asociado al algoritmo de consenso y permiten medir el desempeño de cada sistema en particular. También se debe resaltar que los cálculos de ese estudio fueron realizados sobre la versión 1 de Ethereum, la cual usa como algoritmo de consenso *PoW*. De acuerdo con lo anterior se describen tiempos de aprobación de bloque Ethereum e IoTA respectivamente: *15s y "configurable"*.

Si bien en la evaluación de desempeño realizada en este trabajo, la métrica seleccionada y versiones de la TRD son distintas, es importante comparar los resultados ya que pueden ser significativos para futuros trabajos. La latencia calculada en este estudio incluye el tiempo de envío de la solicitud por parte de la aplicación Gateway, el tiempo de ejecución de la función por el contrato inteligente en la máquina virtual de Ethereum y finalmente el tiempo de aprobación de la transacción y creación del bloque. En este trabajo se utilizó la versión 2 la emplea como algoritmo de consenso *PoS.* La implementación de IoTA se llevó

a cabo utilizando la máquina virtual de Ethereum como habilitante para la ejecución de los contratos inteligentes, pero con la red de IoTA.

Los resultados de esta comparación pueden ser analizados usando la **Tabla 3-3,** donde se hace evidente la mejora en tiempos que experimenta Ethereum entre versiones muy probablemente derivado del cambio del algoritmo de consenso. A pesar de que la latencia calculada en este estudio incluye más pasos del proceso y por tanto se espera un tiempo mayor, el resultado es un tiempo promedio menor que el calculado en (Jabed Morshed Chowdhury et al., s/f).

Los valores de latencia promedio calculados tanto para Ethereum como para IoTA representan nuevos resultados que podrían ser usados para actualizar y complementar lo propuesto por (Jabed Morshed Chowdhury et al., s/f). Esto enriquecería la comprensión y evaluación de estas tecnologías en futuras investigaciones.

Tabla 3-3 Cuadro comparativo de los resultados obtenidos

	Tiempo de aprobación de transacción (Jabed Morshed Chowdhury et al., s/f)	**Latencia de acuerdo con Ecuación 3 1**
Ethereum	**15s** *Versión 1 – Algoritmo Consenso PoW*	**12.6s** *Versión 2 – Algoritmo Consenso PoS*
IoTA	**"Configurable"**	**2.35s** *Usando la MVE*

4. Conclusiones

Para alcanzar los objetivos establecidos en este trabajo final, se elaboró un primer capítulo que comprende una descripción detallada de los principales conceptos relacionados con los recursos de energía distribuida, las microrredes, los sistemas de energía transactiva y las tecnologías de registro distribuido. La caracterización realizada de las TRD y de los sistemas de energía transactiva permitió determinar los criterios de selección de las TRD y los requerimientos funcionales de la aplicación descentralizada en la etapa de diseño del prototipo del sistema de energía transactiva distribuida.

La elaboración de una arquitectura que organizó los artefactos involucrados en la implementación alineándolos de acuerdo con su objetivo, ubicación y afinidad técnica en diferentes capas, facilitó la determinación de los criterios de diseño. La separación de la información y funcionalidades asociadas a la microrred de aquellas asociadas al intercambio económico simplificó la implementación de los contratos inteligentes y permitió determinar la forma de interrelacionar las capas mediante la conexión entre contratos inteligentes y las aplicaciones descentralizadas (capa de borde y de la capa de aplicación). En el modelo de datos, se determinó el uso de direcciones de billetera digital para identificar a los prosumidores, lo cual facilitó la autenticación en aplicación y permitió mantener la confidencialidad de la información del prosumidor. Así mismo, el uso de direcciones de billetera digital para identificar a los medidores permitió que cada prosumidor pueda tener a su cargo varios medidores asociados a cuentas propias, a las cuales se les puede designar un presupuesto limitado para el pago de expensas de las transacciones.

Con una aplicación en la capa de borde cuyos requerimientos de ejecución son ligeros y su funcionamiento dinámico a partir del paso de argumentos por la línea de comandos, se logró la posibilidad de que ésta pueda ser instalada en dispositivos cuyos recursos son limitados como en una Raspberry PI. Además, su dockerización facilitó el escalamiento de las pruebas para la evaluación del desempeño. Sin embargo, en esta aplicación aún

existen retos para el manejo de la seguridad de las cuentas usadas para firmar las transacciones por la exposición de claves privadas en la línea de comandos.

La aplicación descentralizada fue codificada respetando la descomposición funcional propuesta en el diseño lo que favoreció la estructuración de su navegación y el cumplimiento de los requerimientos de usuario. La codificación y despliegue de la aplicación con un diseño equivalente a nivel funcional, pero con distinta apariencia y parámetros del mercado, redujeron los tiempos de desarrollo de una aplicación para cada una de las TRD evaluadas.

La ejecución de las pruebas automáticas en una comunidad de 10 prosumidores durante un período de 5h para 10 medidores reportando valores de energía cada 5 minutos permitió la recolección de información que pudo ser usada para el cálculo de estadísticas de latencia en transacciones de escritura para las redes de prueba de Ethereum e IoTA. El análisis de los resultados permitió concluir que la latencia promedio hora de IoTA es aproximadamente 5 veces menor que la de Ethereum. Además, se identificó que Ethereum podría llegar a tener latencias hasta de 45 segundos, mientras que IoTa llega a máximo a 18 segundos. Estos valores pueden ser utilizados para evaluar la factibilidad de uso de las redes Ethereum e IoTA en sistemas de energía transactiva distribuida en comunidades locales reales.

La comparación de los resultados obtenidos con un estudio previo presenta nuevos datos que podrían ser utilizados para actualizar, complementar y enriquecer la comprensión y evaluación de estas tecnologías en futuras investigaciones.

La implementación y evaluación del prototipo de energía transactiva, demuestra que es posible la integración directa de dispositivos de campo como medidores de energía a una red TRD para el uso de la información en un mercado de energía. Sin embargo, dado que esta evaluación fue realizada mediante un ambiente simulado, aún quedan retos por resolver en la integración de medidores inteligentes, que no podrían haberse hecho evidentes durante este trabajo: implementaciones de los buses de campo para integración de medidores, los dispositivos de borde requeridos para la ejecución de aplicaciones Gateway que integre los medidores a la TRD, interoperabilidad con la implementación del protocolo por cuenta del fabricante del medidor, entre otros.

5. Trabajos futuros

Debido a los diversos aspectos que involucra el diseño de un prototipo de sistema de energía transactiva distribuida, durante el desarrollo de este trabajo se identificaron algunos aspectos que quedaron por fuera del alcance de este trabajo pero que podrían ser tenidos en cuenta en proyectos futuros.

Este prototipo podría servir como base para la implementación de un caso de prueba en una microrred real, lo que permitiría validar aspectos como los tiempos de respuesta, la integración y la interoperabilidad de diferentes protocolos de comunicación según los medidores disponibles en el campo (DNP 3.0, DLMS COSEM, IEC 61850, entre otros). Además, el prototipo podría ser evaluado con otras TRD y también en una red privada, considerando otras métricas de desempeño como la tasa de éxito, la latencia, la tasa de transferencia efectiva, el consumo de recursos y la cantidad de transacciones por segundo. Esto permitiría identificar la combinación más eficiente para la implementación del sistema en una comunidad local

El prototipo de sistema de energía transactiva distribuida podría mejorarse mediante la integración de información a través de oráculos con otros datos. Por ejemplo, debido a la influencia de las condiciones climáticas en el mercado local, podrían integrarse pronósticos del clima y datos del valor de la energía en el mercado regulado para facilitar la toma de decisiones de precios por parte del prosumidor. Además, podría considerarse la inclusión de información sobre el tipo de fuente de energía asociada a cada medidor (eólica, solar, etc.), lo que permitiría a los usuarios seleccionar la fuente que prefieran. Si al sistema se le incluye la capacidad de almacenar datos de tendencia, podría ofrecer a los usuarios gráficas dinámicas del comportamiento de la red y del mercado de energía. Por último, se podrían implementar mecanismos de emisión automática de certificados de origen de la energía, los cuales podrían ser utilizados por los clientes para demostrar su compromiso medioambiental.

Anexo 1: Codificación de los Contratos Inteligentes

Los contratos inteligentes utilizados en este trabajo son EnergyTransact.sol y Microgrid.sol:

EnergyTransact.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.18;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "./Microgrid.sol";


contract EnergyTransact is ERC20 {

    // Valor del token
    uint256 public tokenValue;
    // Usar el contrato MicroGrid
    Microgrid public microgrid;
    // energia que ha comprado un prosumer
     mapping (address => uint256) public prosumerTotalBuyed;


    constructor(uint256 _tokenAmmount, uint256 _tokenValue) ERC20('Energy Market', 'NRG'
) {
        _mint (address(this), _tokenAmmount);
        microgrid = new Microgrid() ;
        tokenValue=_tokenValue*10**12 wei;
    }

    // Precio de los tokens
    function tokenPrice ( uint256 _numTokens ) internal view returns (uint256){
        return _numTokens*(tokenValue);
    }

        // Precio de los tokens
    function setTokenValue ( uint256 _value ) public {
        tokenValue=_value*10**12 wei;
    }

    // Balance de tokens del smart contract
    function balanceTokens() public view returns (uint256){
        return balanceOf(address(this));
    }
```

```
    function balanceEthers() public view returns (uint256){
        return address(this).balance;
    }

    // Comprar tokens
    function buyTokens(uint256 _numTokens) public payable{
        // si el usuario no esta registrado, registrar
        uint256 cost = tokenPrice(_numTokens);
        require(msg.value >= cost, 'No hay dinero suficiente');
        //Valida tokens ERC-20 disponibles
         uint256 balance = balanceTokens();
        require(_numTokens <= balance, 'No hay tokens suficientes');
        //Devoluelve el valor excedente en la compra
        uint256 returnValue = msg.value-cost;
        //Pago
        payable(msg.sender).transfer(returnValue);
        // Entrega de los tokensT
        _transfer(address(this),msg.sender,_numTokens);
    }
    // Comprar tokens
    function returnTokens (uint256 _numTokens) public payable {
        require(_numTokens > 0, 'El valor de la devolucion no es valido');
        require(address(this).balance >= _numTokens,'No hay dinero suficiente');
        _transfer(msg.sender, address(this),_numTokens);
        uint256 returnValue=tokenPrice(_numTokens);
        payable(msg.sender).transfer(returnValue);
    }

    // Comprar energía
    function buyEnergy(address _meterAddress, uint256 _amount) public payable {
        require(_amount > 0);
        require(_meterAddress != address(0),'No existe el medidor' );
        uint256 numTokens=microgrid.reserving(_meterAddress,_amount);
        address _prosumer= microgrid.meterProsumer(_meterAddress);
        require(_prosumer != address(0),'No existe el prosumidor');
        require(_prosumer!=msg.sender, 'Intenta comprar al mismo prosumidor');
        require(balanceOf( msg.sender) >= numTokens,'No hay dinero suficiente');

        _transfer(msg.sender, _prosumer,numTokens);

        microgrid.delivering(_meterAddress,_amount);
        prosumerTotalBuyed[msg.sender]=_amount;
    }
}
```

Microgrid.sol

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.18;

contract Microgrid {

    struct Measures {
        bool active;
        uint256 rate;
        uint256 energyConsumed;
        uint256 energyProduced;
        uint256 energyAvailable;
        uint256 energyDelivered;
        uint256 energyReserved;
    }

    mapping (address => Measures) public meterMeasures;
    mapping (address => address) public meterProsumer;
    mapping (address => address[]) public prosumerMeters;

    address[] public meterList;

    uint256 public totalMeters;

    function createMeter(address _meterAddress, uint256 _rate) public {
        require (_rate > 0);
        require (_meterAddress != address(0), 'Verificar direccion');
        require (meterMeasures[_meterAddress].active==false,'Medidor existente');
        meterProsumer[_meterAddress]=msg.sender;
        meterMeasures[_meterAddress] = Measures(true,_rate,0,0,0,0,0);
        meterList.push(_meterAddress);
        prosumerMeters[msg.sender].push(_meterAddress);
        totalMeters += 1;
    }

    function disableMeter(address _meterAddress) public {
        require (_meterAddress != address(0), 'Verificar direccion');
        meterMeasures[_meterAddress].active = false;
    }

    function enableMeter(address _meterAddress) public {
        require (_meterAddress != address(0), 'Verificar direccion');
        meterMeasures[_meterAddress].active = true;
    }
```

```
    function setRate(address _meterAddress,uint256 _rate) public {
        require (_meterAddress != address(0), 'Verificar direccion');
        meterMeasures[_meterAddress].rate = _rate;
    }


    function meterReport( address _meterAddress, uint256 _energyConsumed, uint256
_energyProduced) public {
        require (_energyConsumed >=0 && _energyProduced >=0 );
        require (_meterAddress != address(0), 'Verificar direccion');
        if (_energyConsumed!=meterMeasures[_meterAddress].energyConsumed){
            meterMeasures[_meterAddress].energyConsumed = _energyConsumed;
        }
        if (_energyProduced != meterMeasures[_meterAddress].energyProduced){
            meterMeasures[_meterAddress].energyProduced = _energyProduced;
        }
        uint256      _energyAvailable=_energyProduced      -      _energyConsumed      -
meterMeasures[_meterAddress].energyDelivered;
        if (_energyAvailable != meterMeasures[_meterAddress].energyAvailable){
            meterMeasures[_meterAddress].energyAvailable =_energyAvailable;
        }


    }

    function reserving(address _meterAddress,uint256 _amount) external returns (uint256)
{
        require (_amount > 0);
        require (_meterAddress != address(0), 'Verificar direccion');
        require(meterMeasures[_meterAddress].energyAvailable   >=   _amount,   'No   hay
suficiente energia disponible');
        meterMeasures[_meterAddress].energyReserved += _amount;
        meterMeasures[_meterAddress].energyAvailable -= _amount;
        return _amount*meterMeasures[_meterAddress].rate;
    }

    function delivering(address _meterAddress,uint256 _amount) external {
        require (_amount > 0);
        require (_meterAddress != address(0), 'Verificar direccion');
        require(meterMeasures[_meterAddress].energyAvailable   >=   _amount,   'No   hay
suficiente energia disponible');
        meterMeasures[_meterAddress].energyDelivered += _amount;
        meterMeasures[_meterAddress].energyReserved -= _amount;
    }

    function rejecting(address _meterAddress,uint256 _amount) external {
```

```
        require (_amount > 0);
        require (_meterAddress != address(0));
        require(meterMeasures[_meterAddress].energyAvailable < _amount, 'No hay suficiente
energia disponible');
        meterMeasures[_meterAddress].energyReserved -= _amount;
    }

    function getProsumerMeters() public view returns (address[] memory){
        return prosumerMeters[msg.sender];
    }
    function getAllMeters() public view returns (address[] memory){
        return meterList;
    }

}
```

Anexo 2: Repositorios de codificación

Todo el código utilizado en la implementación del prototipo puede ser consultado en los siguientes repositorios:

La codificación del Gateway puede ser consultada en el repositorio de GitHub:

https://github.com/lrbecerrab/gateway

La codificación de la aplicación descentralizada que se ejecuta en la red de pruebas de Ethereum (*Sepolia*) puede ser consultada en el repositorio de GitHub:

https://github.com/lrbecerrab/dappethereum

La codificación de la aplicación descentralizada que se ejecuta en la red de pruebas de *IoTA* (*Shimmer*) puede ser consultada en el repositorio de GitHub:

https://github.com/lrbecerrab/dapp*IoTA*

Anexo 3: Despliegues y uso de la Dapp

El prototipo estructuró las opciones del menú para agrupar vistas de acuerdo con el tipo de información global del usuario y con el fin de estructurar la ubicación de las funciones que permiten el cumplimento de los requerimientos de usuario propuestos en el diseño.

Barra de navegación

La barra de navegación es desplegada permanentemente y permite el uso de botones de acceso rápido a las distintas páginas que conforman el prototipo. La barra de navegación se muestra en la **Figura A3-0-1**. En ella están disponibles las siguientes opciones: Welcome, Financial, Market, Account y el botón de enlace a la cuenta del prosumidor

Figura A3-0-1 Barra de navegación

Panel de información del prosumidor:

Independientemente de la página que se esté desplegando, siempre es posible ver la información general del prosumidor (ver **Figura A3-0-2**):

- ***MY ACCOUNT***: El número de cuenta del Prosumidor actual
- ***COIN***: El token/moneda del mercado, es decir NRG.
- ***EXCHANGE RATE (ETH/NRG):*** Tasa de cambio actual entre Ethereum y NRG.
- ***BALANCE:*** Cantidad de NRG que posee el prosumidor actual.

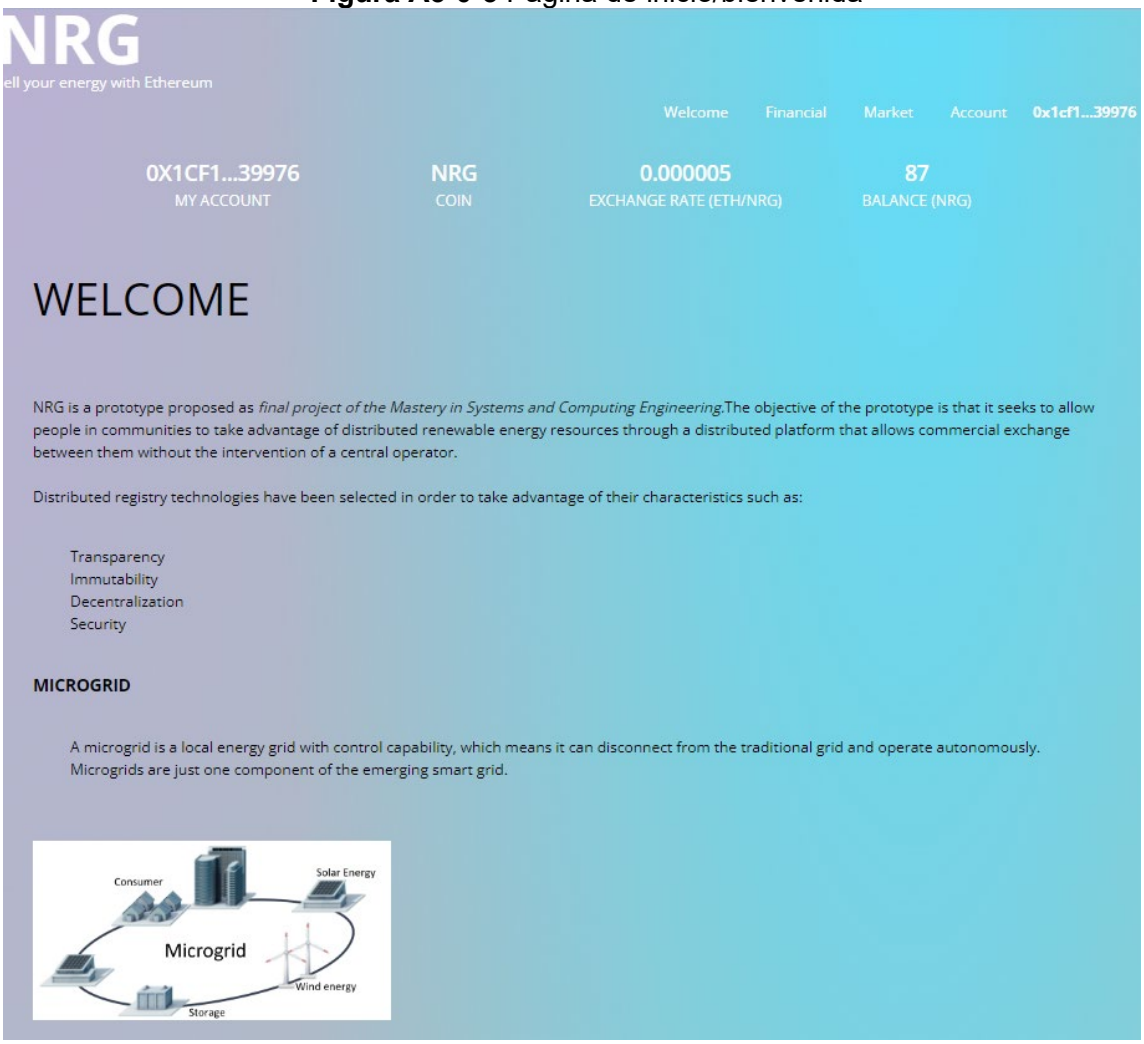
Figura A3-0-2 Panel de información general del prosumidor

Welcome:

Esta opción del menú permite al usuario acceder a la página que contiene breve resumen del proyecto (ver **Figura A3-0-3**). En este resumen se anexa parte de la diagramación del

proyecto para facilitar el entendimiento del mismo. Allí el usuario encuentra el enlace al repositorio *Github* que le permite reproducir la aplicación para hacer otro tipo de pruebas.

Figura A3-0-3 Página de inicio/bienvenida



Financial

Esta opción del menú permite al usuario acceder a la página en el que se muestran todos los datos financieros del mercado (ver **Figura A3-0-4**): tasa de cambio entre ETH/NRG, Cantidad de tokens disponibles en el mercado, cantidad de ETH que ha logrado recaudar el mercado. Así mismo permite efectuar varias operaciones: cambiar la tasa de intercambio de ETH/NRG, comprar NRG (ver **Figura A3-0-5**), devolver NRG.

Figura A3-0-4 Página de datos financieros

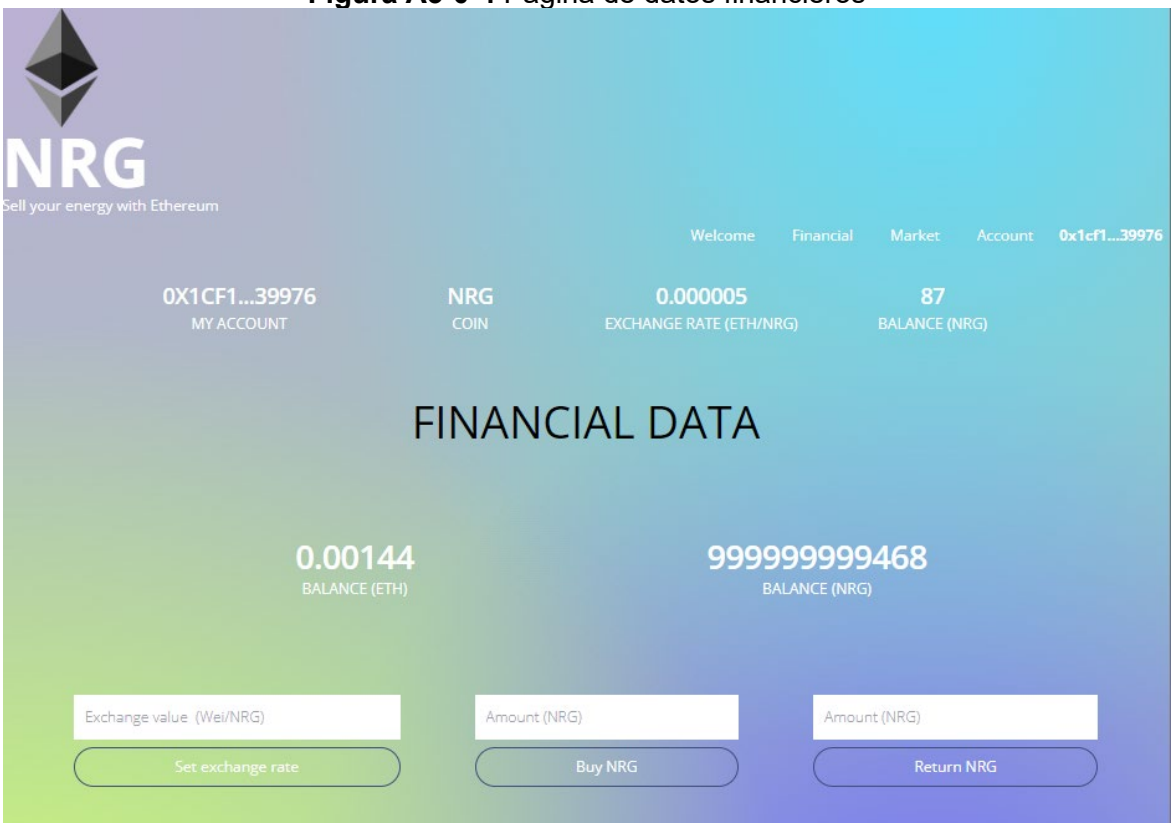
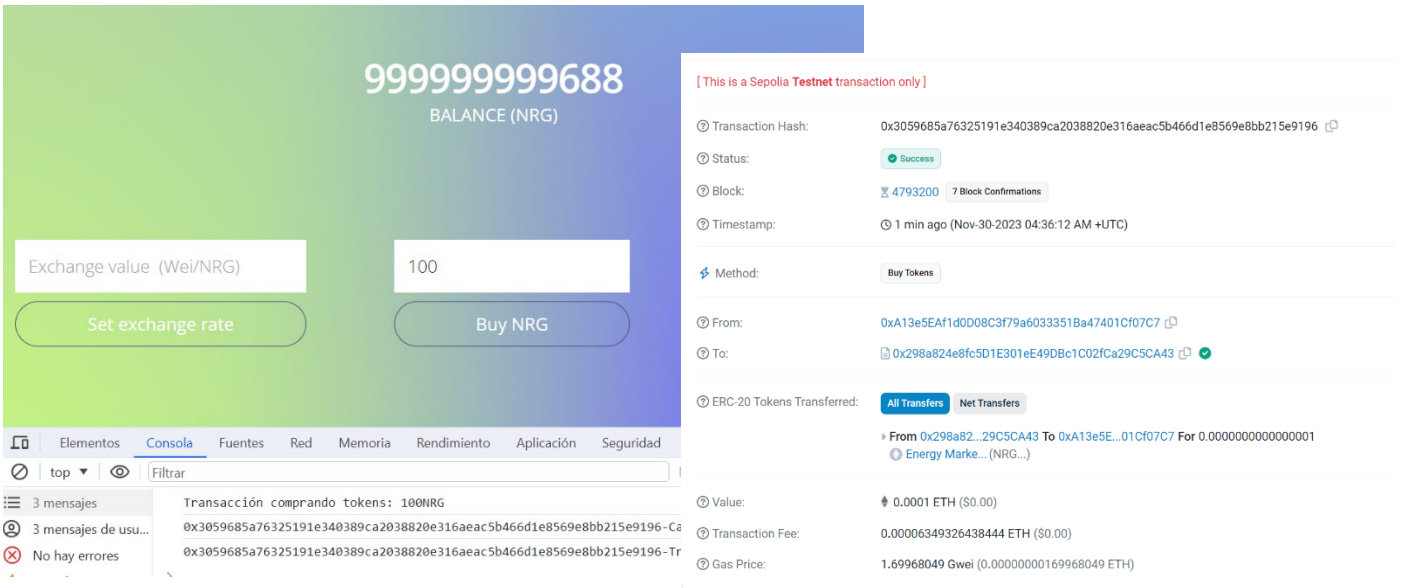


Figura A3-0-5 Transacción de compra de tokens

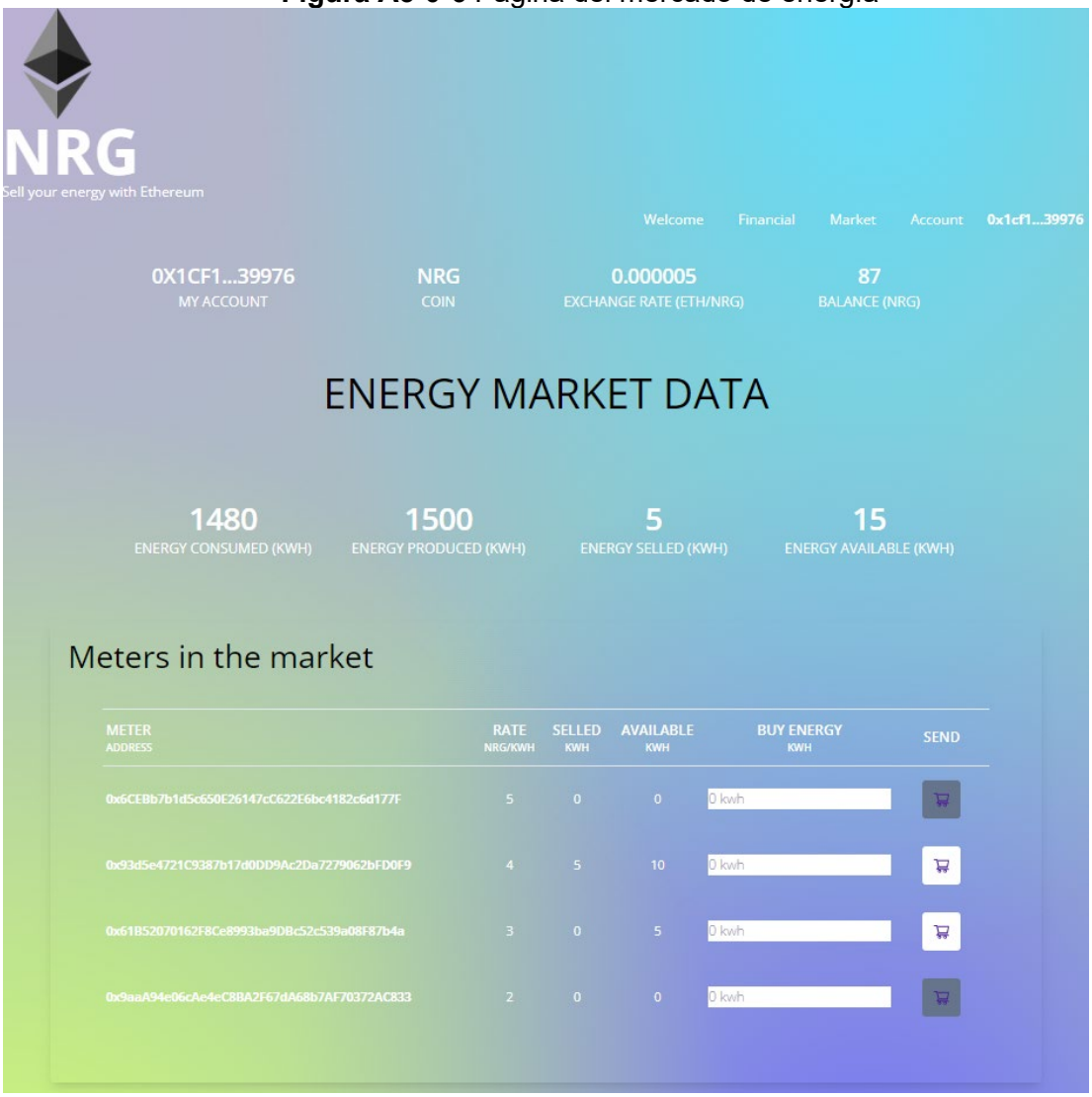


Market

Esta opción del menú permite al usuario acceder a la página en el que se muestran los datos de consumo, producción y disponibilidad de toda la energía del mercado (ver **Figura A3-0-6**).

Así mismo, lista los medidores que se encuentran vinculados al mercado con la información que está siendo reportada en línea de la energía consumida y generada y permite comprar la energía que se encuentra disponible a un precio específico (rate). El prosumidor puede comprar energía a otros prosumidores siempre que la tengan disponible.

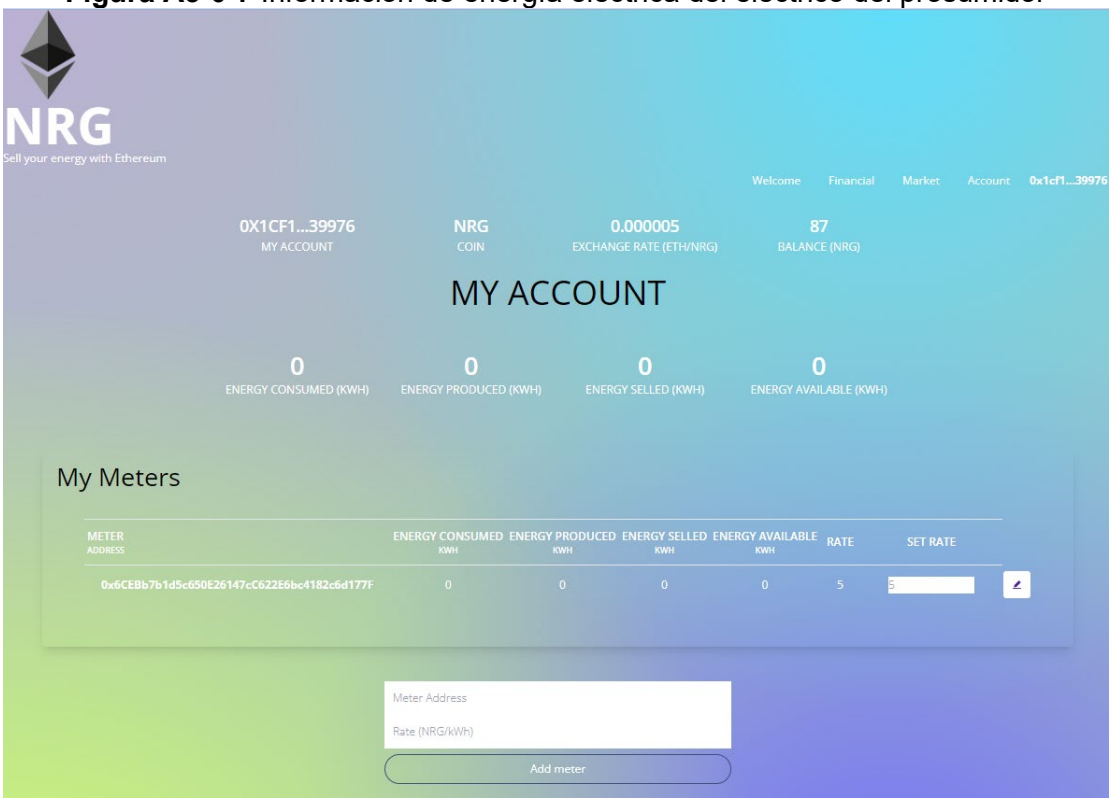
Figura A3-0-6 Página del mercado de energía



Account

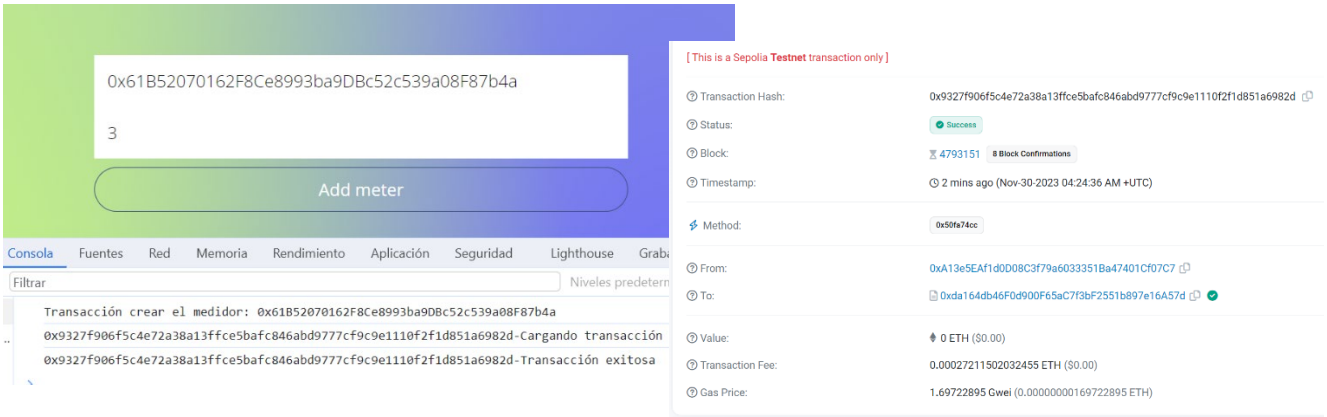
Esta opción del menú permite al usuario acceder a la página en la que el prosumidor puede ver los medidores que tiene en el mercado y así mismo la información que está siendo reportada en cuanto a energía consumida y producida, (ver **Figura A3-0-7**)

Figura A3-0-7 Información de energía eléctrica del eléctrico del prosumidor



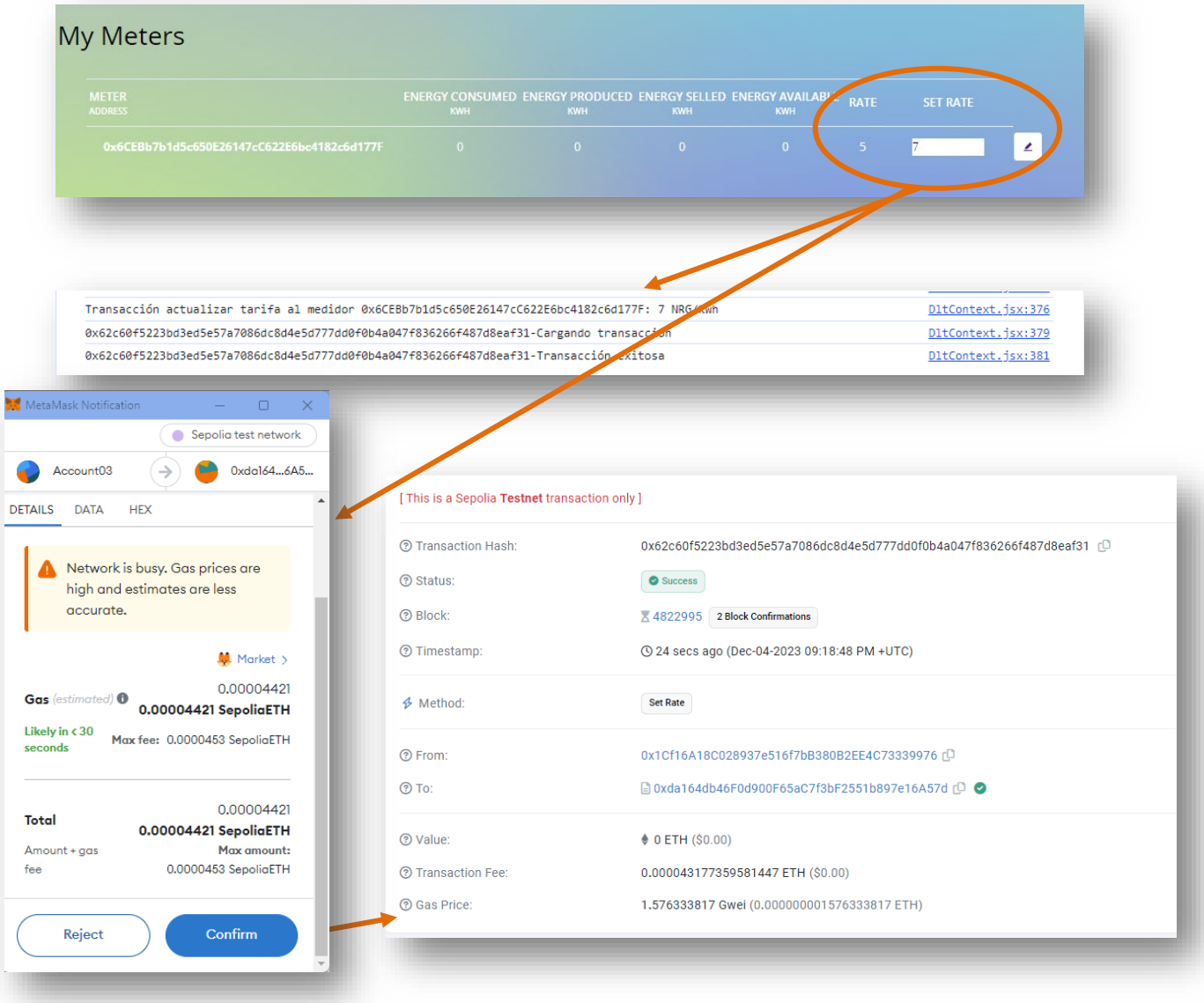
El prosumidor puede crear medidores mediante el uso de un código de cuenta de la billetera digital, como se puede ver en la **Figura A3-0-8**

Figura A3-0-8 Transacción crear medidor



Finalmente, puede cambiar el precio de la energía que ha asignado a cada medidor en su creación de acuerdo como lo describe la **Figura A3-0-9**.

Figura A3-0-9 Ajuste de la tarifa NRG/kwh para un medidor



Anexo 4: Transacciones ejecutadas en la preparación del mercado

Todas las transacciones que se tabulan en este anexo pueden ser consultadas en las páginas disponibles en respectivas redes de prueba:

Ethereum (Sepolia): https://sepolia.etherscan.io/

IoTA (Shimmer): https://explorer.evm.testnet.shimmer.network/

Creación de medidores para cada prosumidor en cada una de las redes de prueba

Prosumidor	Medidor	Transacción con la que fue creado en Ethereum	Transacción con la que fue creado en *IoTA*
Account01	Met01	0xab15e8a2d4108032511d938766055afe88c996f766f98a1f9123689505557fe5	0x0369cdcabf63bfd0d609e9592a60b561a7bdde4b59847b6767ac3ae10c03a41b
Account02	Met02	0x4b9b9450f37ed335c027923a21f09a6cca45a965c735c3f7b3d67b807c548f16	0x589a265ac27bd37505c30f49ee56987ccb6b0d1b691b4b8259ce8e807bb57bff
Account03	Met03	0xddfdf32b07f039029477185fbe1aec7ad61e4f03c8a232d9a751107df2729ca9	0x51a296515da02967049444b0267e41036f9a9fd6ce5ec3cf6c323370f69b8142
Account04	Met04	0x60ab53ac0f7e03765909e345331b831cd77164c67a36d64da226aeab4f670a64	0x2af999e2d7fdad45dfc0b512e5765f714a407eca35d7fd3c65601c8f5d5d230c
Account05	Met05	0xd49aa8ba87cd9edb0d2cdf9d74cfbb280e9898632bb19811cc597609dd055776	0x3d2678a6941d668f192c8913a1034e9c69730f2fd7e46a1a28183b2d1e390b7f
Account06	Met06	0xeae2a97bb28d1375859920fbce95116f1020316c885835b9fc33994e9243d02b	0x242d1c281ffd29f043458045f9d2787fa80acb88b2a88b7eb5cf34460f7a8489
Account07	Met07	0x6eedd79e5f6ec412b5be57cc4fbf7e4d854422e956a4a15ae5059b6991a403be	0x072743d2e89003edb2410289ca442e97370823f29888e4cbfc32df8e987b48ed
Account08	Met08	0x37eb7f364252d6ccd6037dbd59807ff71c0541ca5a59689b4ef6b3c08c8cdbb6	0xd575f18c3d6925cbe515fbb8d65a1d2f3ce5bf2201457a82bba2460dc0ecfc26
Account09	Met09	0x4817fc6c047ef938f5511bb5f0f5399532e3dfb2ea5be42058caa7136e615ad4	0x233a1e456b270f41dc87ddd529c1fdb9fa40f3c77447b00a577b9e258677027c
Account10	Met10	0xe97226a1b4000c8bee66341ced876d2a45ea68f9cdc33e6887e6dd3d0d9f8163	0x23e7eb732e5f7bcb80508fe1e439e0eb3230463ae1f6c56fb85b8e205803b78c

Transacciones de compra de tokens para ser utilizados en la comercialización de energía en cada una de las redes de prueba

Prosumidor	Devolver 100 NRG	Devolver 100 NRG
Account01	0x8b888c047cf5a8433b1071391f8f357499044e03dfabc1c6af6f0e89e2c2ae95	0x939786232afafc84ce893c8d00625d583155ca0e9636a9f0a6827f8f4e884ec3
Account02	0xa271391e602e48e4d0aed0c40121b8d8d7846e17aefa23c5a891a0fb9b25bcb3	0xf89e8ecc67d8bde280a110fa88cb7bcec77884a2504bbba40fd25a20adb7b101
Account03	0x1e8d469556853016a91681383ad7d3dc667581bb35978ffa864f60f0655cd3c1	0x341049d1c6329974c49d30e8768f7677a3f6b0f57a6cd13acd5c64c48bc1eed3
Account04	0xb6f2a467d630d9aaa18ae60bf2114f280d8350d38dae4b26d8e7e48b965f9818	0x93f8b1b52cc575953874c63824b4f75c0a86364f383b40de53ef1b4581a1eda2
Account05	0x60bf4b2d8c48c187f6a248d2affeb52eed82c2b1ec38a66dd8f3ef332f7c2540	0x669f70735ec9ccbe20672addc27a78396f77342414246b518805fcfa5a083784
Account06	0xd8a16ef724a7744a09d546eb38d1a8d235077028ac9e904a6d29c48b69f883bc	0xed120a6187b73b25766fafd74af0ede4c46dd2be3dd207b14bf7c5870c68b0d5
Account07	0xf1e7e5cd393f25f1d8ecaf64147426ed9b69a67c4d95bce4319e99586126b8e7	0x548504ce077e3e5f61c096768a871fe372665160e6c1bf9d5e75de9c13719ff5
Account08	0x15d961d171284412bcf220e58aea1a844f25083688b92b622f21e11821a2d489	0x903a71157443ccc4a8b2c938befd68d19324fe898153e612ef91b06fa2443c2b
Account09	0xceef1db2d143b4a455d07ec4471dab52571f69c032742444461e7df32cbca3c4	0x0e276ea72ee1181ebff3f6b2b3a1a9339fff29b87148c6285360750d092e3852
Account10	0x93d0f0fce6e9d183cf964d4ad8f267f3428efd1ab1503286cab274ac62298b6f	0xefb8b35e26b3b5844d23206ea7668188d8d0508760e3c67f030332be45745efc

Transacciones de devolución de tokens para recuperar los valores en ETH / SMR

Prosumidor	Transacción cambiar 100 NRG a ETH	Transacción cambiar 100 NRG a SMR
Account01	0x8b888c047cf5a8433b1071391f8f357499044e03dfabc1c6af6f0e89e2c2ae95	0x939786232afafc84ce893c8d00625d583155ca0e9636a9f0a6827f8f4e884ec3
Account02	0xa271391e602e48e4d0aed0c40121b8d8d7846e17aefa23c5a891a0fb9b25bcb3	0xf89e8ecc67d8bde280a110fa88cb7bcec77884a2504bbba40fd25a20adb7b101
Account03	0x1e8d469556853016a91681383ad7d3dc667581bb35978ffa864f60f0655cd3c1	0x341049d1c6329974c49d30e8768f7677a3f6b0f57a6cd13acd5c64c48bc1eed3
Account04	0xb6f2a467d630d9aaa18ae60bf2114f280d8350d38dae4b26d8e7e48b965f9818	0x93f8b1b52cc575953874c63824b4f75c0a86364f383b40de53ef1b4581a1eda2
Account05	0x60bf4b2d8c48c187f6a248d2affeb52eed82c2b1ec38a66dd8f3ef332f7c2540	0x669f70735ec9ccbe20672addc27a78396f77342414246b518805fcfa5a083784
Account06	0xd8a16ef724a7744a09d546eb38d1a8d235077028ac9e904a6d29c48b69f883bc	0xed120a6187b73b25766fafd74af0ede4c46dd2be3dd207b14bf7c5870c68b0d5
Account07	0xf1e7e5cd393f25f1d8ecaf64147426ed9b69a67c4d95bce4319e99586126b8e7	0x548504ce077e3e5f61c096768a871fe372665160e6c1bf9d5e75de9c13719ff5
Account08	0x15d961d171284412bcf220e58aea1a844f25083688b92b622f21e11821a2d489	0x903a71157443ccc4a8b2c938befd68d19324fe898153e612ef91b06fa2443c2b
Account09	0xceef1db2d143b4a455d07ec4471dab52571f69c032742444461e7df32cbca3c4	0x0e276ea72ee1181ebff3f6b2b3a1a9339fff29b87148c6285360750d092e3852
Account10	0x93d0f0fce6e9d183cf964d4ad8f267f3428efd1ab1503286cab274ac62298b6f	0xefb8b35e26b3b5844d23206ea7668188d8d0508760e3c67f030332be45745efc

Bibliografía

Abdella, J., Tari, Z., Anwar, A., Mahmood, A., & Han, F. (2021). An Architecture and Performance Evaluation of Blockchain-Based Peer-to-Peer Energy Trading. *IEEE Transactions on Smart Grid*, *12*(4), 3364–3378. https://doi.org/10.1109/TSG.2021.3056147

Ali, S. S., & Choi, B. J. (2020). State-of-the-art artificial intelligence techniques for distributed smart grids: A review. *Electronics (Switzerland)*, *9*(6), 1–28. https://doi.org/10.3390/electronics9061030

Andoni, M., Robu, V., Flynn, D., Abram, S., Geach, D., Jenkins, D., McCallum, P., & Peacock, A. (2019). Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable and Sustainable Energy Reviews*, *100*, 143–174. https://doi.org/10.1016/j.rser.2018.10.014

Antal, C., Cioara, T., Anghel, I., Antal, M., & Salomie, I. (2021). Distributed ledger technology review and decentralized applications development guidelines. En *Future Internet* (Vol. 13, Número 3, pp. 1–32). MDPI AG. https://doi.org/10.3390/fi13030062

Bertone, F., Caragnano, G., Simonov, M., Goga, K., & Terzo, O. (2020). A Classification of Distributed Ledger Technology Usages in the Context of Transactive Energy Control Operations. *Advances in Intelligent Systems and Computing*, *993*, 876–885. https://doi.org/10.1007/978-3-030-22354-0_81

Buterin, V. (2015). *A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM*.

Chen, X., Nakada, R., Nguyen, K., & Sekiya, H. (2021). A Comparison of Distributed Ledger Technologies in IoT: IOTA versus Ethereum. *Proceedings of ISCIT 2021: 2021 20th International Symposium on Communications and Information Technologies: Quest for Quality of Life and Smart City*, 182–187. https://doi.org/10.1109/ISCIT52804.2021.9590601

CorDapp Design Language (CDL) overview - R3 Documentation. (s/f). Recuperado el 26 de enero de 2024, de https://docs.r3.com/en/tools/cdl/cdl-overview.html

Cullen, A., Ferraro, P., King, C., & Shorten, R. (2020). On the Resilience of DAG-Based Distributed Ledgers in IoT Applications. *IEEE Internet of Things Journal*, *7*(8), 7112–7122. https://doi.org/10.1109/JIOT.2020.2983401

Dong, Z., Zheng, E., Choon, Y., & Zomaya, A. Y. (2019). DAGBENCH: A performance evaluation framework for DAG distributed ledgers. *IEEE International Conference on Cloud Computing, CLOUD*, *2019-July*, 264–271. https://doi.org/10.1109/CLOUD.2019.00053

Dr, W., & Baliga, A. (2020). *Understanding Blockchain Consensus Models*.

Energy - United Nations Sustainable Development. (s/f). Recuperado el 15 de enero de 2022, de https://www.un.org/sustainabledevelopment/energy/

Energy Production and Consumption - Our World in Data. (s/f). Recuperado el 12 de junio de 2021, de https://ourworldindata.org/energy-production-consumption

ENERGY TRANSITION TOWARDS THE ACHIEVEMENT OF SDG 7 AND NET-ZERO EMISSIONS Secretariat of the High-level Dialogue on Energy 2021 Division for Sustainable Development Goals Department of Economic and Social Affairs. (s/f). Recuperado el 16 de diciembre de 2021, de https://www.un.org/en/conferences/energy2021/about

Factory Contract – Blockchain Patterns. (s/f). Recuperado el 27 de enero de 2024, de https://research.csiro.au/blockchainpatterns/general-patterns/contract-structural-patterns/factory-contract/

Fan, C., Ghaemi, S., Khazaei, H., & Musilek, P. (2020). Performance Evaluation of Blockchain Systems: A Systematic Survey. *IEEE Access*, *8*, 126927–126950. https://doi.org/10.1109/ACCESS.2020.3006078

Geun Song, J., seon Kang, E., Woo Shin, H., Wook Jang, J., Smart, J. A., & Blockchain, E. (2021). *A Smart Contract-Based P2P Energy Trading System with Dynamic Pricing on Ethereum Blockchain Contract-Based P2P Energy Trading System with Dynamic Pricing on*. https://doi.org/10.3390/s21061985

Giotitsas, C., Pazaitis, A., & Kostakis, V. (2015). A peer-to-peer approach to energy production. *Technology in Society*, *42*, 28–38. https://doi.org/10.1016/j.techsoc.2015.02.002

Górski, T., & Bednarski, J. (2020). Modeling of distributed ledger deployment view. *International Journal of Electronics and Telecommunications*, *66*(4), 619–625. https://doi.org/10.24425-ijet.2020.134020/743

Hayes, B. P., Thakur, S., & Breslin, J. G. (2020). Co-simulation of electricity distribution networks and peer to peer energy trading platforms. *International Journal of Electrical Power and Energy Systems*, *115*. https://doi.org/10.1016/j.ijepes.2019.105419

Jabed Morshed Chowdhury, M., Ferdous, S., Biswas, K., Chowdhury, N., M Kayes, A. S., Alazab, M., & Watters, P. (s/f). *A Comparative Analysis of Distributed Ledger Technology Platforms*. https://doi.org/10.1109/ACCESS.2019.2953729

JavaScript Environment Requirements – React. (s/f). Recuperado el 7 de diciembre de 2023, de https://legacy.reactjs.org/docs/javascript-environment-requirements.html

Kirpes, B., Mengelkamp, E., Schaal, G., & Weinhardt, C. (2019). Design of a microgrid local energy market on a blockchain-based information system. *IT - Information Technology*, *61*(2–3), 87–99. https://doi.org/10.1515/ITIT-2019-0012/MACHINEREADABLECITATION/RIS

Marnay, C., Chatzivasileiadis, S., Abbey, C., Iravani, R., Joos, G., Lombardi, P., Mancarella, P., & Von Appen, J. (2015). Microgrid evolution roadmap. *Proceedings - 2015 International Symposium on Smart Electric Distribution Systems and Technologies, EDST 2015*, 139–144. https://doi.org/10.1109/SEDST.2015.7315197

Mengelkamp, E., Gärttner, J., Rock, K., Kessler, S., Orsini, L., & Weinhardt, C. (2018). Designing microgrid energy markets: A case study: The Brooklyn Microgrid. *Applied Energy*, *210*, 870–880. https://doi.org/10.1016/j.apenergy.2017.06.054

Mengelkamp, E., Notheisen, B., Beer, C., Dauer, D., & Weinhardt, C. (2018). A blockchain-based smart grid: towards sustainable local energy markets. *Computer Science - Research and Development*, *33*(1–2), 207–214. https://doi.org/10.1007/s00450-017-0360-9

Miglani, A., Kumar, N., Chamola, V., & Zeadally, S. (2020). Blockchain for Internet of Energy management: Review, solutions, and challenges. *Computer Communications*, *151*, 395–418.

Muhanji, S. O., Flint, A. E., & Farid, A. M. (2019). eIoT: The development of the energy internet of things in energy infrastructure. En *eIoT: The Development of the Energy Internet of Things in Energy Infrastructure*. https://doi.org/10.1007/978-3-030-10427-6

OpenZeppelin | Contracts. (s/f). Recuperado el 8 de diciembre de 2023, de https://www.openzeppelin.com/contracts

Pedro, J., & Lopes, A. (2023). *Exploração de algoritmos de consenso no Quorum*. https://recipp.ipp.pt/handle/10400.22/23439

Pervez, H., Muneeb, M., Irfan, M. U., & Ul Haq, I. (2019). A Comparative Analysis of DAG-Based Blockchain Architectures. *ICOSST 2018 - 2018 International Conference on Open Source Systems and Technologies, Proceedings*, 27–34. https://doi.org/10.1109/ICOSST.2018.8632193

Proof-of-stake (PoS) | ethereum.org. (s/f). Recuperado el 26 de enero de 2024, de https://ethereum.org/developers/docs/consensus-mechanisms/pos

Siano, P., De Marco, G., Rolan, A., & Loia, V. (2019). A Survey and Evaluation of the Potentials of Distributed Ledger Technology for Peer-to-Peer Transactive Energy Exchanges in Local Energy Markets. *IEEE Systems Journal*, *13*(3), 3454–3466. https://doi.org/10.1109/JSYST.2019.2903172

Skowronski, R. (2017). On the applicability of the GRIDNET protocol to Smart Grid environments. *undefined*, *2018-January*, 200–206. https://doi.org/10.1109/SMARTGRIDCOMM.2017.8340700

Sousa, T., Soares, T., Pinson, P., Moret, F., Baroche, T., & Sorin, E. (2019). Peer-to-peer and community-based markets: A comprehensive review. *Renewable and Sustainable Energy Reviews*, *104*, 367–378. https://doi.org/10.1016/j.rser.2019.01.036

The Architecture of a Web 3.0 application. (s/f). Recuperado el 29 de noviembre de 2022, de https://www.preethikasireddy.com/post/the-architecture-of-a-web-3-0-application

Transactive Energy Systems Research, Development and Deployment Roadmap Prepared by the GridWise ® Architecture Council. (2018). www.gridwiseac.org

Vieira, G., & Zhang, J. (2021). Peer-to-peer energy trading in a microgrid leveraged by smart contracts. *Renewable and Sustainable Energy Reviews*, *143*. https://doi.org/10.1016/j.rser.2021.110900

Wohrer, M., Zdun, U., & Rinderle-Ma, S. (2021). Architecture Design of Blockchain-Based Applications. *2021 3rd Conference on Blockchain Research and Applications for Innovative Networks and Services, BRAINS 2021*, 173–180. https://doi.org/10.1109/BRAINS52497.2021.9569813

Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., Pautasso, C., & Rimba, P. (2017). *A Taxonomy of Blockchain-Based Systems for Architecture Design*. https://doi.org/10.1109/ICSA.2017.33

Zhang, C., Wu, J., Long, C., & Cheng, M. (2017). Review of Existing Peer-to-Peer Energy Trading Projects. *Energy Procedia*, *105*, 2563–2568. https://doi.org/10.1016/J.EGYPRO.2017.03.737

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

Zia, M. F. M. F., Elbouchikhi, E., Benbouzid, M., & Guerrero, J. M. J. M. (2019). Microgrid Transactive Energy Systems: A Perspective on Design, Technologies, and Energy Markets. *IECON Proceedings (Industrial Electronics Conference)*, *2019-Octob*, 5795–5800. https://doi.org/10.1109/IECON.2019.8926947

Zia, M. F., Member, S., Benbouzid, M., Elbouchikhi, E., Member, S., Muyeen, S. M., Techato, K., & Guerrero, J. M. (s/f). *Microgrid Transactive Energy: Review, Architectures, Distributed Ledger Technologies, and Market Analysis*. https://doi.org/10.1109/ACCESS.2020.2968402