ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN INGENIERÍA INDUSTRIAL

TRABAJO DE FIN DE MÁSTER

POTENCIAL DE LA TECNOLOGÍA BLOCKCHAIN EN EL MERCADO ELÉCTRICO

Autor: *Daniel García Guzmán*

Directores:

Íñigo García de Mata

Ignacio Herrero Gallego

Francisco Martín Martínez

Madrid, Julio de 2018

AUTORIZACIÓN PARA LA DIGITALIZACIÓN, DEPÓSITO Y DIVULGACIÓN EN RED DE PROYECTOS FIN DE GRADO, FIN DE MÁSTER, TESINAS O MEMORIAS DE BACHILLERATO

1º. Declaración de la autoría y acreditación de la misma.

El autor D. Daniel García Guzmán

DECLARA ser el titular de los derechos de propiedad intelectual de la obra:

"Potencial de la tecnología Blockchain en el mercado eléctrico", que ésta es una obra original, y que ostenta la condición de autor en el sentido que otorga la Ley de Propiedad Intelectual.

2º. Objeto y fines de la cesión.

Con el fin de dar la máxima difusión a la obra citada a través del Repositorio institucional de la Universidad, el autor **CEDE** a la Universidad Pontificia Comillas, de forma gratuita y no exclusiva, por el máximo plazo legal y con ámbito universal, los derechos de digitalización, de archivo, de reproducción, de distribución y de comunicación pública, incluido el derecho de puesta a disposición electrónica, tal y como se describen en la Ley de Propiedad Intelectual. El derecho de transformación se cede a los únicos efectos de lo dispuesto en la letra a) del apartado siguiente.

3º. Condiciones de la cesión y acceso

Sin perjuicio de la titularidad de la obra, que sigue correspondiendo a su autor, la cesión de derechos contemplada en esta licencia habilita para:

a) Transformarla con el fin de adaptarla a cualquier tecnología que permita incorporarla a internet y hacerla accesible; incorporar metadatos para realizar el registro de la obra e incorporar "marcas de agua" o cualquier otro sistema de seguridad o de protección.
b) Reproducirla en un soporte digital para su incorporación a una base de datos electrónica, incluyendo el derecho de reproducir y almacenar la obra en servidores, a los efectos de garantizar su seguridad, conservación y preservar el formato.
c) Comunicarla, por defecto, a través de un archivo institucional abierto, accesible de modo libre y gratuito a través de internet.
d) Cualquier otra forma de acceso (restringido, embargado, cerrado) deberá solicitarse expresamente y obedecer a causas justificadas.
e) Asignar por defecto a estos trabajos una licencia Creative Commons.
f) Asignar por defecto a estos trabajos un HANDLE (URL *persistente).*

4º. Derechos del autor.

El autor, en tanto que titular de una obra tiene derecho a:

a) Que la Universidad identifique claramente su nombre como autor de la misma
b) Comunicar y dar publicidad a la obra en la versión que ceda y en otras posteriores a través de cualquier medio.
c) Solicitar la retirada de la obra del repositorio por causa justificada.
d) Recibir notificación fehaciente de cualquier reclamación que puedan formular terceras personas en relación con la obra y, en particular, de reclamaciones relativas a los derechos de propiedad intelectual sobre ella.

5º. Deberes del autor.

El autor se compromete a:

a) Garantizar que el compromiso que adquiere mediante el presente escrito no infringe ningún derecho de terceros, ya sean de propiedad industrial, intelectual o cualquier otro.
b) Garantizar que el contenido de las obras no atenta contra los derechos al honor, a la intimidad y a la imagen de terceros.
c) Asumir toda reclamación o responsabilidad, incluyendo las indemnizaciones por daños, que pudieran ejercitarse contra la Universidad por terceros que vieran infringidos sus derechos e intereses a causa de la cesión.

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 04 de Julio de 2018

ACEPTA

Fdo 

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN INGENIERÍA INDUSTRIAL

TRABAJO DE FIN DE MÁSTER

POTENCIAL DE LA TECNOLOGÍA BLOCKCHAIN EN EL MERCADO ELÉCTRICO

Autor: *Daniel García Guzmán*

Directores:

Íñigo García de Mata

Ignacio Herrero Gallego

Francisco Martín Martínez

Madrid, Julio de 2018

POTENCIAL DE LA TECNOLOGÍA BLOCKCHAIN EN EL MERCADO ELÉCTRICO

Autor: García Guzmán, Daniel.

Directores: García de Mata, Íñigo; Herrero Gallego, Ignacio; Martín Martínez, Francisco.

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

1. Estado del arte

Es evidente e indiscutible que Blockchain ha irrumpido con fuerza en el panorama tecnológico en los años recientes, atrayendo las miradas de instituciones de distintos sectores. Si restringimos el ámbito de estudio al mercado eléctrico, el interés por la tecnología Blockchain se ha materializado en la proliferación de un elevado número de proyectos que podemos clasificar en diferentes líneas de desarrollo:

1) comercio mayorista y minorista,
2) optimización de procesos del mercado (B2B),
3) promoción de la energía renovable,
4) desarrollo de plataformas de uso general para el sector.

Del estudio de los desarrollos más destacados se puede extraer, primero, que existen aún numerosos problemas para la implantación real de este tipo de proyectos, entre los cuales destacan la ausencia de regulación, los fuertes costes derivados de la utilización de redes públicas y el escepticismo que frecuentemente caracteriza a los clientes potenciales. En segundo lugar, se observa que no existe un criterio unánime en torno a cómo se debe utilizar la tecnología, aspecto que se refleja en la disparidad de posturas adoptadas en decisiones clave (por ejemplo, decidir si la plataforma ha de ser pública o privada para una aplicación concreta). Además, no están claras las ventajas e inconvenientes que Blockchain ofrece frente a los sistemas utilizados actualmente en el sector eléctrico.

2. Caso de estudio

En este TFM se aborda el diseño conceptual y desarrollo de un prototipo de mercado de derivados de electricidad basado en Blockchain, a fin de evaluar eficazmente el potencial de la tecnología en este campo. La selección del mercado de derivados como pieza de estudio viene motivada por dos factores:

1) la casación continua es computacionalmente menos costosa que la casación de una subasta;
2) es posible contemplar una liquidación del contrato exclusivamente financiera.

Los objetivos planteados exigen el diseño de una arquitectura basada en *smart contracts* que permita reproducir de manera simplificada los procesos básicos del mercado, así como la programación de diferentes simulaciones que validen el modelo de mercado y proporcionen la información que se desea analizar. Para ello se utilizan distintas herramientas de código abierto, gracias a las cuales es posible trabajar en un entorno Blockchain de prueba que simula el funcionamiento de la red Ethereum.

3. Breve descripción del modelo

El prototipo de mercado de derivados se estructura en torno a tres contratos principales (piezas de código) que interactúan entre sí:

1) *Token*. Representación del valor única empleada en todo el sistema.
2) Mercado. Representación del sistema de casación.
3) Ticket. Representación de asignación de ofertas casadas.

Esquema simplificado del funcionamiento del sistema

Conceptualmente el funcionamiento del sistema es simple: los usuarios envían sus ofertas a través del contrato mercado, que registra la información y comprueba el resultado de casación de manera automática en tiempo real. Cuando se produce la

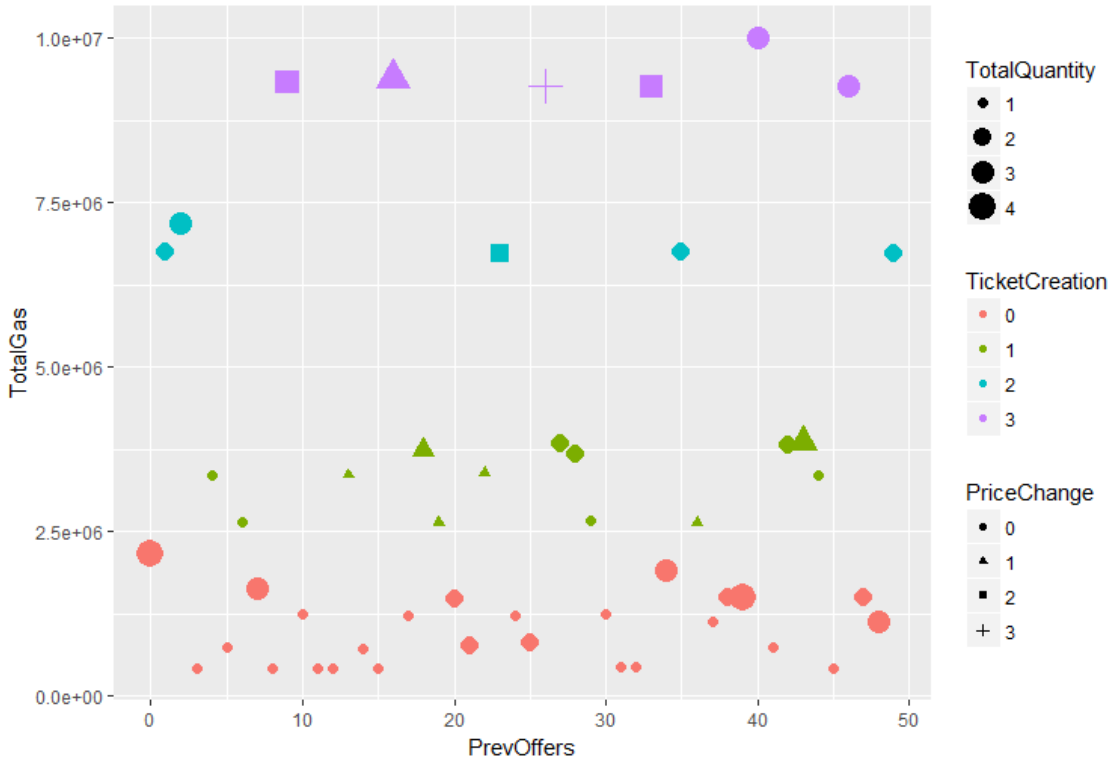
casación de dos ofertas, el contrato mercado despliega sobre la red dos contratos ticket, de compra y venta respectivamente. Los contratos ticket son los encargados de actualizar periódicamente el balance del usuario con cada señal de precio volcada sobre la red, para liquidar las cantidades resultantes al finalizar el período de entrega estipulado. Cuando el ticket ha cumplido su misión, se autodestruye y sus funciones dejan de estar disponibles. Tanto el contrato mercado como los contratos ticket utilizan los métodos del contrato *token* para transferir valor entre las cuentas.

El algoritmo de casación utilizado es capaz de corregir el precio de la oferta entrante cuando ésta pueda casar a un precio más favorable. Además, el prototipo incorpora algunas funciones avanzadas para mejorar su usabilidad, tales como la posibilidad de cancelar ofertas o vender el contrato ticket una vez generado (sólo su propietario).

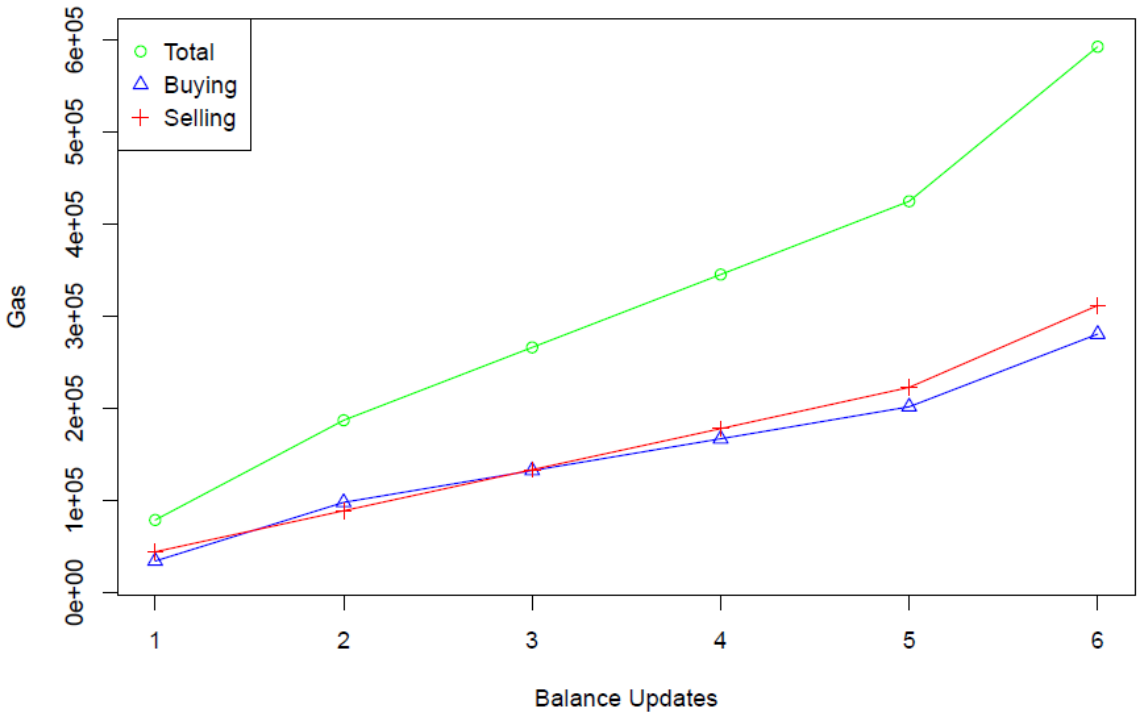
4. Resultados y conclusiones

Tras la realización de las pruebas pertinentes, los contratos que conforman el modelo han demostrado responder adecuadamente, por lo que se valida el funcionamiento del prototipo de mercado de derivados en el entorno de simulación conforme a las especificaciones. Con ello se certifica la viabilidad técnica de la integración de mercados financieros (como el mercado de derivados) en entornos Blockchain.

Por otro lado, se ha simulado el comportamiento del mercado ante el envío de ofertas en escenarios de diferente magnitud, así como el proceso de liquidación en diferentes situaciones. Los resultados obtenidos proporcionan evidencias sólidas del elevado coste computacional (medido en consumo de gas) de la integración del sistema en Blockchain.



Simulación del coste en gas asociado al envío de ofertas



Simulación del coste en gas asociado a la actualización del balance del ticket

Además, de manera específica podemos extraer las siguientes conclusiones acerca de la viabilidad económica de soluciones de este tipo:

- el mayor esfuerzo computacional del sistema se corresponde con el proceso de casación, lo cual indica que la opción más eficiente de cara a una aplicación real sería desacoplar esta parte del proceso y ejecutarlo de manera externa;
- el desarrollo integral en redes públicas se demuestra ineficiente.

Descartada la opción de utilización de redes públicas, son dos las opciones que tienen cabida desde el punto de vista económico:

4) desarrollo de una red privada *ad-hoc*;
5) desarrollo de una red permisionada sobre una plataforma de propósito general como Ethereum.

Conclusiones más precisas acerca de la aplicabilidad de esta tecnología en el ámbito del mercado mayorista requieren de un análisis más exhaustivo.

POTENTIAL OF THE BLOCKCHAIN TECHNOLOGY IN WHOLESALE ELECTRICITY MARKETS

Author: García Guzmán, Daniel.

Directors: García de Mata, Íñigo; Herrero Gallego, Ignacio; Martín Martínez, Francisco.

Collaborating Entity: ICAI – Universidad Pontificia Comillas.

PROJECT SUMMARY

1. State of the art

It is evident and indisputable that Blockchain has strongly irrupted in the technological scene in recent years, catching the attention of different sectors' institutions. If we limit the field of study to energy markets, the interest about Blockchain technology has turn into the spread of a high number of projects which can be classified in various development lines:

1) wholesale and retail markets,
2) market processes optimization (B2B),
3) renewable energy promotion,
4) general use platforms development for energy sector.

After studying the most relevant related projects several conclusions have been extracted: first, that there are still many problems linked to the real implantation of this type of projects, including regulatory issues, public platforms utilization-derived high costs and the skepticism (frequently observed) of potential clients. Secondly, it can be seen there is no common criteria respecting how this technology must be used, which is clearly reflected in the diversity of opinions about key points (for instance, given a certain application, deciding if the platform must be public or private). Furthermore, advantages and disadvantages on blockchain systems over currently used ones in energy sector are not quite clear.

2. Case study

Here we tackle the conceptual design and development of a blockchain-based power derivatives market prototype, in order to evaluate the technology potential in this field. Power derivatives market selection as particular subject of enquiry is motivated by two main factors:

1) matching up of continuous markets is computationally less expensive than performing daily auctions;
2) considering exclusively financial liquidations is possible.

Purposed objectives require the design of a smart contract-based architecture which let us reproduce the basic market processes in a simplified manner, as well as developing different simulations which validate the market model and yield the information to be

analyzed. In order to do this, several open source tools are used, thanks to which working in a blockchain test environment is possible.

3. Brief model description

The power derivatives market prototype is structured around three main contracts (code pieces) which interact between them:

1) *Token*. Value representation.
2) Market. Matching system representation.
3) Ticket. Representation of matched offers assignment.

Simplified diagram of system operation

The system operation is conceptually simple: users must send their offers through market contract, which registers the information and checks the matching process result automatically in real time. When matching of two offers occurs, the market contract deploys two ticket contracts, buying and selling respectively. Ticket contracts are in charge of updating periodically the user balance when receiving a new price signal, thus they can liquidate final quantities at the end of the stablished delivery period. When the ticket has accomplished its mission, self-destruction happens, and its functions are not

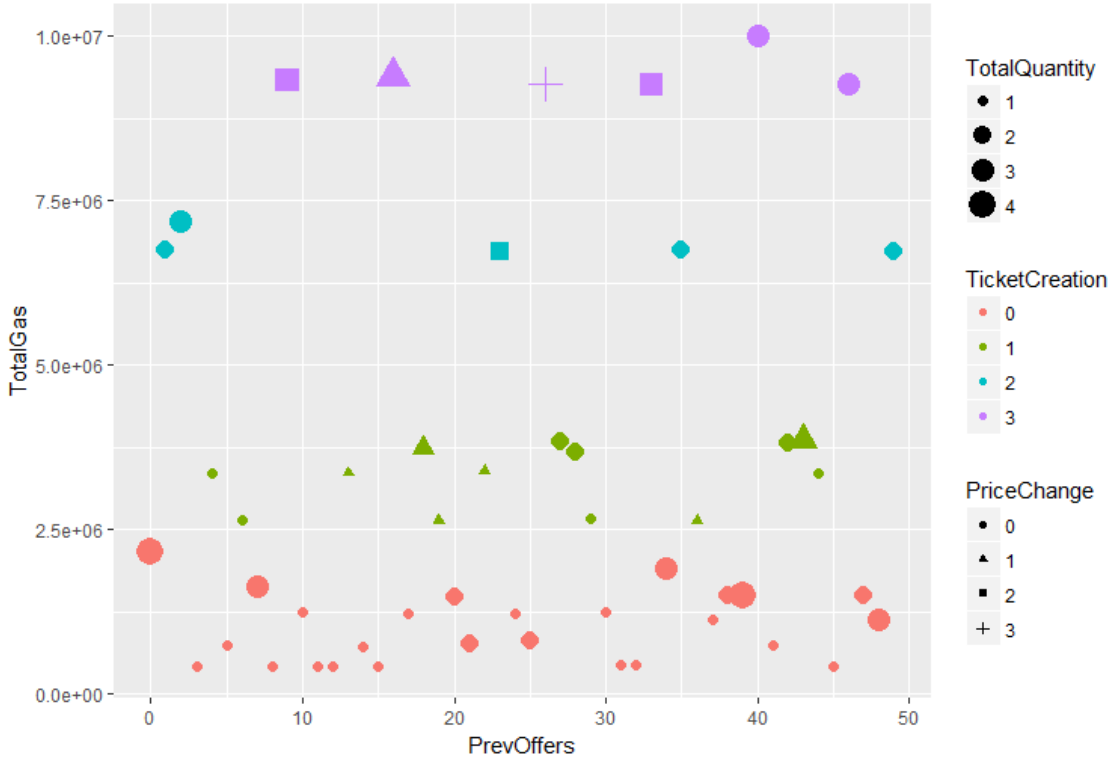
available any more. Both market and tickets use token contract methods to transfer value between accounts.

Matching algorithm is able to correct the entrant offer price when matching at better price is possible. Furthermore, prototype incorporates some advanced functions to improve its usability, such as the possibility of cancelling offers or selling the ticket contract once is generated (only its owner).

4. Results and conclusions

After developing the corresponding tests, market prototype contracts have proved they perform as expected, so the model is validated according to specifications in the test environment. Thus, technical viability of financial markets (such as the derivatives market) integration in blockchain environments is certified.

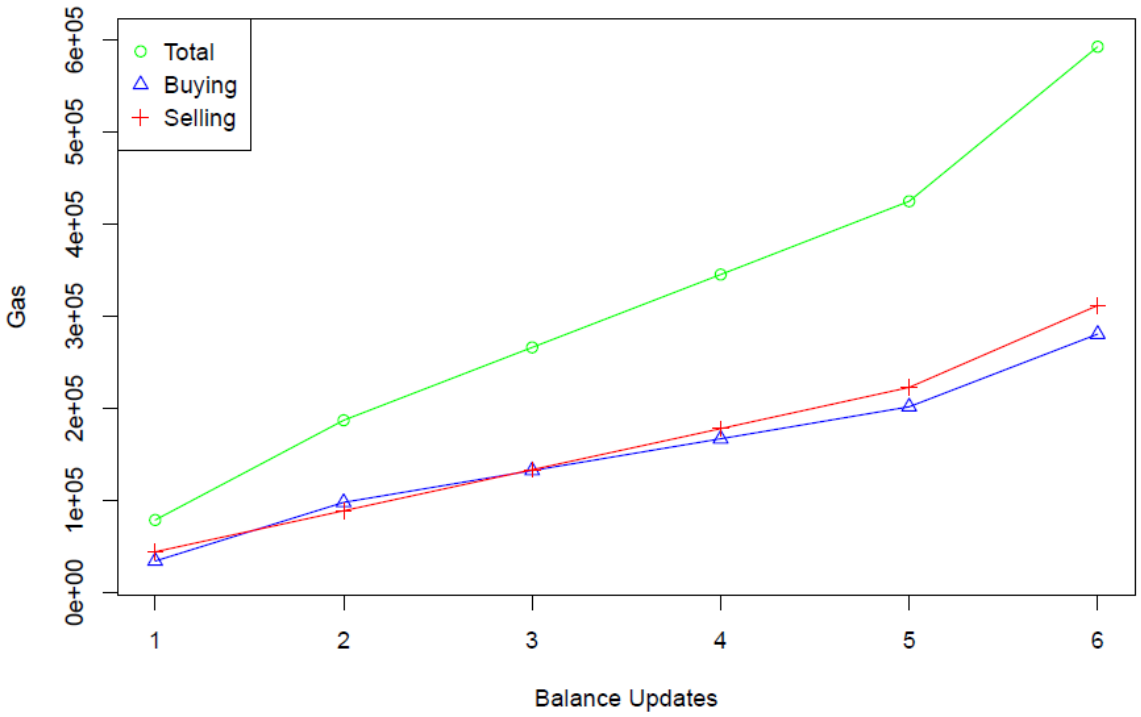
On the other hand, market behavior against the offer sending have been simulated in various-size scenarios, as well as market to market operations in different situations. Results obtained provide strong evidence of the high computational cost (measured in gas used) of the system integration in blockchain.



Simulation of gas cost related to sending offers

Furthermore, specifically we can extract the following conclusions about the economic viability of this type of solutions:

- system highest computational effort corresponds to the matching process, which points the most efficient option to develop a real application would be separate this part of the process and execute it externally;
- full development in public blockchains is proved inefficient.



Simulation of gas cost related to ticket balance update

Removing the option of using public blockchains, there are two reasonable options from an economic prospective:

1) development of an *ad-hoc* private platform;
2) development of a permissioned platform, built on public general-purpose blockchain (Ethereum, for instance).

More accurate conclusions about the blockchain applicability in wholesale markets field require further analysis.

Contenido

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

El creciente interés que manifiestan compañías de diversa índole por la tecnología Blockchain junto con el ineludible apoyo de los medios de comunicación (especialmente Internet) han catapultado este concepto a la primera plana del panorama tecnológico mundial. Aunque el ritmo es pausado, parece que el paso del tiempo está permitiendo disipar la niebla de expectativas e interrogantes que desde un primer momento se cernía sobre el universo Blockchain, y el número de empresas que están apostando por la tecnología es ya elevado.

En el ámbito de los mercados eléctricos, iniciativas como la microrred de Brooklyn [1] invitan a plantearse qué papel puede llegar a desempeñar la tecnología Blockchain en un sector que afronta nuevos retos. A corto plazo, diferentes fuentes sostienen que su implementación puede suponer una simplificación significativa en numerosos procesos de mercado [2] [3]. A medio-largo plazo, el comercio energético a través de redes P2P (*peer-to-peer*) puede facilitar un mayor protagonismo de la generación distribuida, donde

nuevos mercados más accesibles incentiven por sí mismos la participación de pequeños consumidores.

Hoy por hoy, la realidad es que, pese al enorme potencial de desarrollo que sugieren diversos autores, las posibilidades de Blockchain en el sector eléctrico se encuentran en su mayoría inexploradas y existen multitud de cuestiones que requieren de una investigación más exhaustiva. De hecho, podemos destacar que:

1) No hay una clasificación definida de las aplicaciones de Blockchain al sector eléctrico.
2) Dado un área concreta de aplicación, no están claras las ventajas/inconvenientes respecto a los sistemas existentes.

El objetivo de este TFM es evaluar la aplicabilidad de la tecnología Blockchain en un área muy concreta del sector eléctrico: el mercado mayorista. Para aportar evidencias sólidas al respecto, se lleva a cabo la especificación e implementación de un prototipo de mercado de derivados, desarrollo central del proyecto, detallado en los capítulos cuarto y quinto de este texto.

Previamente, es necesario analizar el estado del arte e introducir ciertas nociones del mercado eléctrico, temas tratados en los capítulos segundo y tercero respectivamente. Los resultados obtenidos se recogen en el capítulo sexto, mientras que el capítulo séptimo se ha reservado para las conclusiones del estudio.

Capítulo 2

Estado de la cuestión

Es significativo el número de desarrollos relacionados con Blockchain que han surgido de manera reciente. El objetivo de este capítulo es hacer un recorrido por las iniciativas más destacadas, haciendo hincapié en aquéllas que tienen el *trading* energético por objeto directo de aplicación o bien están intrínsecamente relacionados.

Se ha creído conveniente exponer en términos generales la base tecnológica de Blockchain en primer lugar. Por ello, en 2.1 se propone una definición del concepto, se hace un breve recorrido por los aspectos más importantes que caracterizan esta tecnología y se recogen las plataformas más destacadas actualmente. En la sección 2.2 se presentan algunos ejemplos representativos de otros sectores para finalmente, en 2.3, centrar el punto de mira en el sector eléctrico. Al final del capítulo se incluye una clasificación de algunas de las iniciativas más relevantes en este ámbito.

2.1 Fundamentos de la tecnología Blockchain

Aunque la denominación *"Blockchain"* es posterior, la definición originaria de este término se puede buscar en el *paper* original de Bitcoin [4], donde se presenta la moneda electrónica como *"una cadena de firmas digitales"*. Con el paso del tiempo, el concepto ha adquirido un sentido más amplio, de forma que cuando se habla de Blockchain es posible hacer referencia a tres cosas distintas muy relacionadas[1]:

1) una criptomoneda,
2) la tecnología en cuanto que fundamento teórico,
3) una red de ordenadores (plataforma) que utiliza dicha tecnología, y que puede desarrollar o no una criptomoneda propia.

Esta sección se centra en lo segundo: el objetivo es describir y analizar el concepto Blockchain como base tecnológica que permite la creación de activos digitales y que necesita de una red de ordenadores para su funcionamiento práctico. Desde el prisma descrito, en este TFM se entiende por **Blockchain** lo siguiente:

> *"**Blockchain** es un registro distribuido de transacciones verificables mediante sistemas de criptografía asimétrica, que almacena la información en forma de bloques conectados entre sí. Cada bloque enlaza con el anterior a través de una función criptográfica, generando la cadena que da nombre al concepto."*

Para analizar realmente cómo funciona, es necesario desgranar y analizar separadamente cada punto de la definición aquí propuesta.

2.1.1 Registro distribuido y no descentralizado

En primer lugar, se dice que las redes Blockchain son distribuidas, en contraposición a las redes centralizadas y descentralizadas, porque cada elemento de la red puede funcionar simultáneamente como emisor y receptor respecto a los demás nodos de la red. Todos los nodos se encuentran interconectados sin necesidad de un servidor intermedio.

[1] Es posible encontrar en la literatura distintas clasificaciones. Algunos autores [47] hablan de Blockchain 1.0, 2.0 y 3.0 para referirse a las criptomonedas, los contratos inteligentes y las aplicaciones basadas en ellos, respectivamente.

A diferencia de otros sistemas, no existen nodos específicos que desempeñen la función de almacenamiento de datos; en lugar de ello, la información se registra en todos los nodos que componen la red. Por ello, Blockchain es un registro distribuido en el cual cada nodo de la red contiene una copia de la cadena de bloques. Dicho de otra forma, Blockchain actúa como un libro mayor distribuido que recoge todas las transacciones ejecutadas hasta la fecha a través de la plataforma.

2.1.2 Criptografía en Blockchain

Los **sistemas de criptografía simétrica,** tradicionalmente empleados para la protección de mensajes confidenciales, se caracterizan por el uso de una única clave, lo que significa que la misma clave utilizada para encriptar el mensaje es la que se utiliza para descifrarlo. El problema de este sistema es que el emisor debe enviar al receptor tanto el mensaje como la clave necesaria para desencriptarlo, y, por tanto, cualquiera que interceptase el par mensaje-clave podría conocer su contenido. Lo que se hace en realidad es depositar la confianza en el canal de distribución: si el medio es inseguro, este sistema de cifrado resulta inútil.

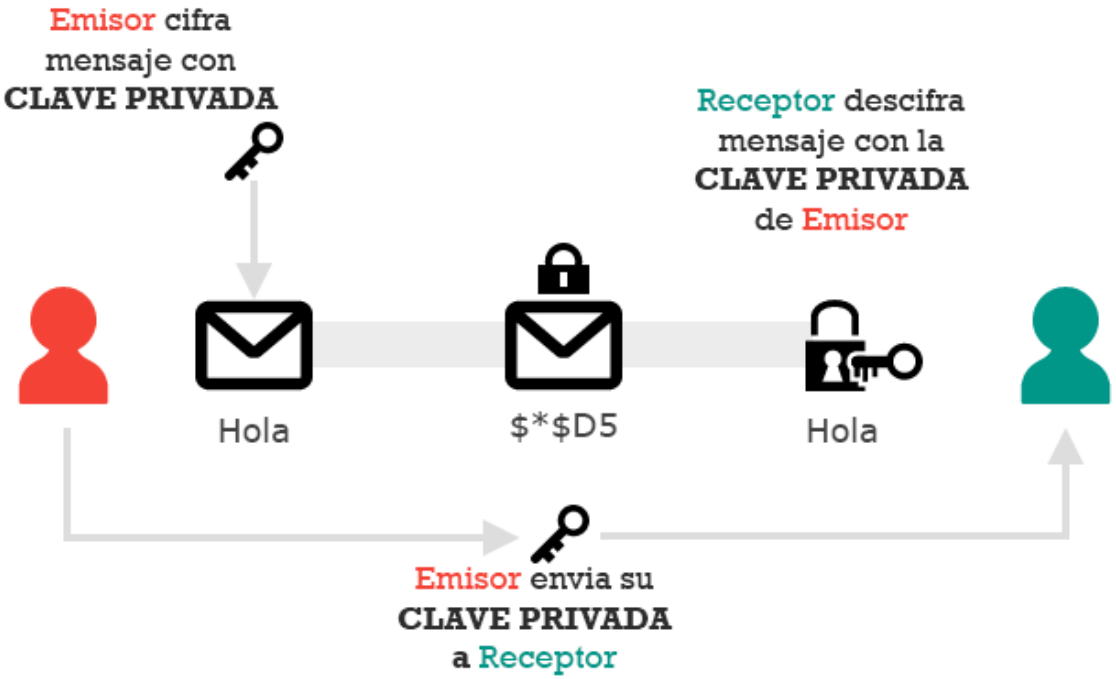


Figura 2.1. Esquema de funcionamiento de un sistema de criptografía simétrica. Fuente: Tipos de criptografía: criptografía simétrica, criptografía asimétrica y criptografía híbrida [6].

Los sistemas de criptografía asimétrica o de clave pública surgen a finales del siglo XX para solventar las limitaciones de la criptografía simétrica [5].

En los **sistemas de criptografía asimétrica**, cada usuario dispone de dos claves: una clave pública, visible para cualquiera, y una clave privada que únicamente él conoce. Para proteger un mensaje, el emisor lo cifra utilizando la clave pública del receptor, de tal forma que sólo puede ser descifrado con la clave privada de éste. Como sólo el receptor conoce su clave privada, nadie más que él puede acceder al contenido del mensaje.

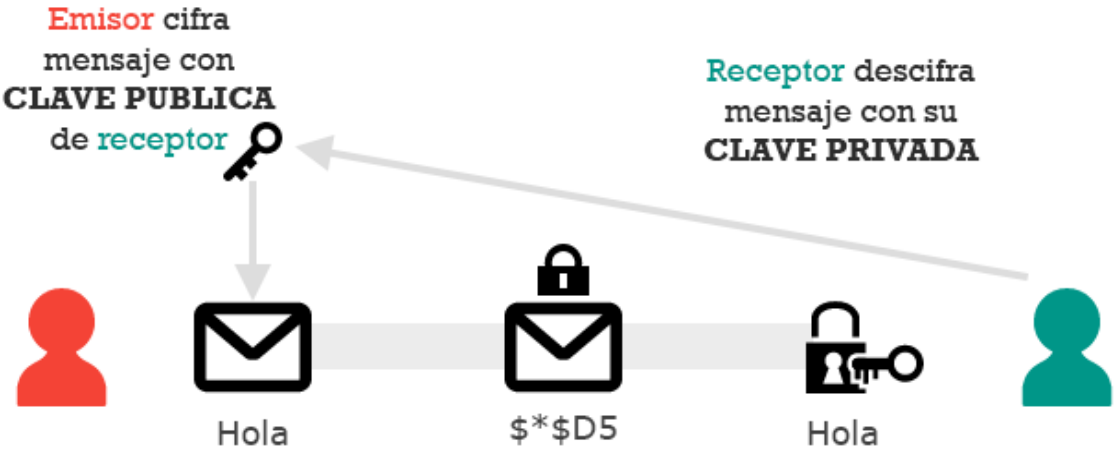


Figura 2.2. Esquema de funcionamiento de un sistema de criptografía asimétrica aplicado al cifrado de mensajes. Fuente: Tipos de criptografía: criptografía simétrica, criptografía asimétrica y criptografía híbrida [6].

En el contexto de Blockchain, el concepto de criptografía asimétrica no se utiliza para encriptar mensajes, sino para verificar la autenticidad de las transacciones ejecutadas mediante lo que se conoce como firma digital. Cuando un usuario ejecuta una transacción, dicha transacción queda automáticamente firmada con su clave privada. A partir de ese momento, cualquiera puede comprobar la autenticidad del mensaje utilizando la clave pública del emisor. De este modo, es posible en primer lugar conocer inequívocamente la cuenta de la que procede el mensaje, y en segundo lugar certificar que no se ha producido ninguna alteración de su contenido.

Por último, cabe mencionar que la robustez del sistema queda supeditada a la capacidad del algoritmo criptográfico. Aunque no se entrará en profundidad, la idea básica es que la función utilizada ha de cumplir con una serie de propiedades para que el sistema sea seguro. Actualmente, el ECDSA (*Elliptic Curve Digital Secure Algorithm*) es el mecanismo utilizado por Bitcoin, Ethereum y la mayor parte de protocolos [7].

2.1.3 Privacidad de la red

Antes de entrar a explicar cómo se añaden los nuevos bloques de información a la cadena, es interesante distinguir entre dos tipos de Blockchain desde el punto de vista de la privacidad:

- En una Blockchain pública (como Bitcoin o Ethereum) no existe autoridad alguna que pueda restringir los derechos de los miembros de la red, por lo que los nodos pueden incorporarse a la red o abandonarla en cualquier momento, así como desempeñar funciones de lectura o escritura.
- En una Blockchain *permisionada* (como Hyperledger), por el contrario, los derechos de lectura y/o escritura se determinan y asignan por parte de un organismo central.

Tal y como se sugiere en [8], desde la perspectiva de Blockchain el permiso de lectura se aplica a cualquier entidad que participe en el proceso de creación de una transacción, o simplemente en la extracción y análisis de información contenida en la cadena de bloques. Por otro lado, el permiso de escritura corresponde a las entidades involucradas en el mecanismo de consenso, concepto que se trata más adelante en esta misma sección, y que está relacionado con el modo en que los bloques se insertan en la cadena.

Es habitual en la literatura referirse a las plataformas permisionadas como privadas, si bien este uso no es del todo correcto. En este TFM, la denominación Blockchain privada se reserva para aquellas plataformas en las cuales ambos derechos de escritura y lectura se encuentren restringidos. Por tanto, se entiende que toda plataforma privada es por definición permisionada, no así a la inversa: si sólo se limitan los derechos de escritura (es decir, el estado del sistema es verificable por cualquier nodo), entonces se considera que la Blockchain es permisionada pública, siguiendo con la propuesta del artículo citado en el párrafo anterior.

2.1.4 Mecanismo de consenso

En su origen, y aunque tal y como se ha descrito en el punto anterior es posible desarrollar plataformas permisionadas, las redes Blockchain tienen un marcado carácter público. No en vano, la idea detrás de la concepción primigenia de Blockchain es la descentralización, esto es, la eliminación de intermediarios. Para conseguir este objetivo, un aspecto fundamental es igualar el rol de todos los participantes en la red.

Si se limita el espectro de análisis dejando fuera la vertiente de índole privada, la descentralización plantea un inconveniente: no existen en el sistema entidades específicas que verifiquen la información, por lo que la validación de los datos inscritos en el registro se ha de lograr mediante un procedimiento alternativo. Este procedimiento no es otra cosa sino un mecanismo de consenso:

> *"El **mecanismo de consenso** es el procedimiento mediante el cual los nodos de la red deben ponerse de acuerdo para determinar qué información es veraz y cuál es fraudulenta."*

El mecanismo de consenso es, por tanto, una herramienta de seguridad que permite evaluar la veracidad de la información que se intenta incorporar a la base de datos distribuida.

En el caso de una Blockchain pública, el mecanismo de consenso tiene una función adicional, que es la base para proporcionar seguridad en un entorno que es por naturaleza inseguro. Para evitar que un individuo intente introducir información fraudulenta en la red y dado que no se dispone de una autoridad central que dictamine el buen o mal comportamiento de dicho participante, el propio sistema debe desincentivar las acciones deshonestas. En terminología de la teoría de juegos, la red se puede entender como un equilibrio de Nash [9]. Ningún jugador tiene incentivos para cambiar individualmente su estrategia (en este caso, comportarse de manera maliciosa) porque de hacerlo el beneficio obtenido sería menor.

Para describir el concepto de consenso y explicar cómo se generan los bloques de la cadena, se va a tomar como referencia el mecanismo de consenso paradigmático, que además es el utilizado actualmente por las cadenas públicas más importantes: la prueba de trabajo (PoW, *Proof of Work*).

2.1.5 Prueba de trabajo

Podemos definir la prueba de trabajo de la siguiente manera:

> *"La **prueba de trabajo** es un esfuerzo computacional que realizan los nodos mineros[2] para encontrar un valor objetivo de características determinadas, el cual servirá para identificar un bloque concreto de la cadena una vez confirmado."*

El resultado de la prueba de trabajo es siempre una clave alfanumérica de longitud fija (*hash*) que se obtiene de aplicar un algoritmo matemático al conjunto de datos del bloque.

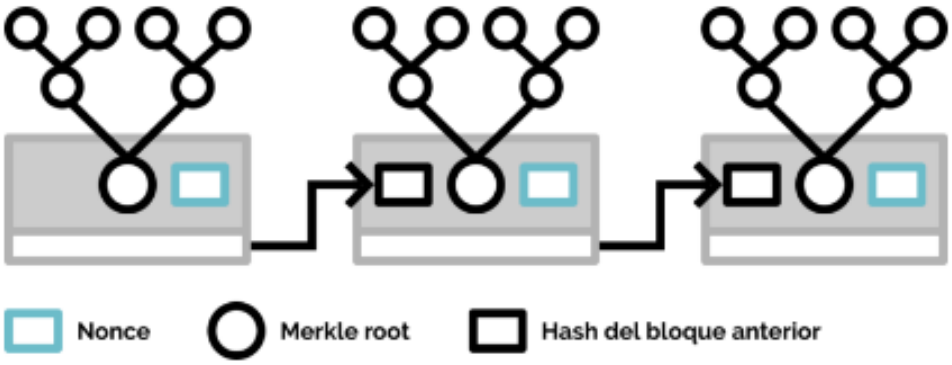


Figura 2.3. Representación simplificada de los elementos contenidos en cada bloque de la cadena. Fuente: Blockchain: la revolución industrial de Internet [9].

Cada bloque de la cadena contiene la siguiente información [10]:

- *Hash* del bloque anterior. Sirve para enlazar la cadena de bloques. Este aspecto es clave para la integridad de la Blockchain; si un agente malicioso intentase modificar un dato registrado en la Blockchain, entonces este cambio alteraría el *hash* del bloque que contuviese la transacción en cuestión, modificando consigo todos los bloques subsiguientes de la cadena. Los *hashes* de los bloques siguientes dejarían de ser por tanto válidos y el ataque a la red sería invalidado[3].
- Transacciones del bloque. Las transacciones que hayan tenido lugar desde el último bloque se agrupan por pares formando lo que se conoce como un árbol de Merkle. Básicamente, se genera un *hash* por cada par de transacciones, éstas se vuelven a emparejar y se repite el

[2] Se denomina así a los nodos que participan en el proceso de cálculo.
[3] Para que un atacante pudiese modificar la información de las transacciones registradas en la Blockchain, tendría que desarrollar una capacidad de cálculo superior al resto de la red entera para tener posibilidades de modificar los datos del bloque que los contiene y reproducir el resto de la cadena. Hoy en día, esto es del todo inviable.

proceso hasta obtener el *hash* raíz (*root hash*) del árbol Merkle. Éste es el valor que el minero utiliza para el cálculo[4].

- *Timestamp*. Cambia cada cierto intervalo temporal, haciendo que el minero tenga que repetir desde cero el proceso de cálculo.
- *Nonce* (*number used once*). Es el contador que el minero incrementa en cada iteración para modificar el *hash* del bloque.

De manera simplificada, el proceso de cálculo es el siguiente [9]:

1) el minero agrupa la información del bloque, le aplica una función criptográfica y obtiene un cierto valor.
2) Para comprobar si el valor calculado es correcto, el algoritmo de autenticación debe verificar que éste reúne unas ciertas condiciones predefinidas.
3) Si el resultado es correcto, el minero ha resuelto la prueba y el proceso ha terminado. En caso contrario, la secuencia se repite desde el principio.

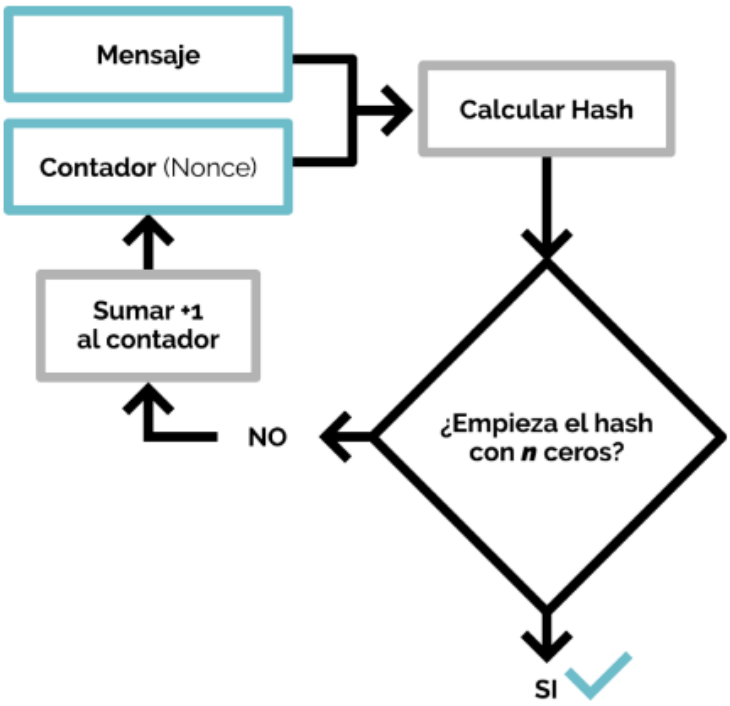


Figura 2.4. Esquema simplificado del proceso de cálculo para encontrar un hash *de dificultad* n *en un algoritmo tipo* hashcash. *Fuente:* Blockchain: la revolución industrial de Internet [9].

[4] Nótese que, gracias a la estructura del árbol de Merkle, un bloque de 1.000 transacciones se mina a la misma velocidad que otro que sólo tenga una, ya que sólo se necesita el *hash* raíz. Cualquier modificación de una de las transacciones del árbol modificará por completo la raíz e invalidará su valor.

En el caso de los algoritmos tipo *hashcash* (tal es el caso de Bitcoin), el *hash* resultante debe contener un cierto número de ceros iniciales. De esta forma, el minero debe calcular repetidamente el *hash* del contenido del bloque modificando el *nonce* en cada iteración hasta dar con un resultado válido, lo que significa resolver la prueba de trabajo. La dificultad de este proceso se puede incrementar variando el número de ceros con los que ha de comenzar el resultado. En este caso, es habitual referirse al *hash* con *n* ceros iniciales como *hash* de dificultad *n*. La Figura 2.4 muestra el flujograma simplificado del proceso de cálculo que típicamente realizan los mineros durante la prueba de trabajo.

De esta explicación se pueden extraer dos conclusiones importantes:

1) que la seguridad de la red depende de su tamaño y
2) que resolver la prueba de trabajo es un proceso costoso y conlleva un elevado consumo energético.

El problema que se plantea es obvio: interesa que la cantidad de nodos mineros posible se encuentren operativos para maximizar la seguridad de la red, y, sin embargo, el proceso de cálculo supone un gran coste, lo cual desincentiva la creación de nuevos nodos. Por este motivo y para que los mineros estén dispuestos a asumir dicho coste, las plataformas que utilizan un algoritmo tipo PoW como mecanismo de consenso deben incorporar algún tipo de recompensa. Este incentivo consiste frecuentemente en premiar al minero que resuelva en primer lugar la prueba de trabajo (lo que se denomina minar un bloque) con una cierta cantidad de la criptodivisa en cuestión (por ejemplo, Bitcoin o *Ether* en caso de Ethereum). Por tanto, añadir un bloque a la cadena genera *tokens*[5], de donde surge el concepto de minería.

Como se puede colegir del párrafo anterior, las criptomonedas surgen de la necesidad de hacer sostenible el mecanismo de la prueba de trabajo. No obstante, también es cierto que este algoritmo se diseña inicialmente para introducir la moneda electrónica (la idea original de Bitcoin), de modo es difícil determinar si fue primero el huevo o la gallina. Lo que sí queda patente es que, en un primer momento, se establece una relación de total dependencia entre plataforma y criptomoneda. Sin embargo, esto no tiene por qué ser así; en redes privadas los mecanismos de consenso pueden no incluir incentivos de este tipo puesto que son, por naturaleza, innecesarios.

[5] Un *token* es cualquier unidad de valor artificial que se registra sobre la Blockchain.

Por tanto, es importante decir que plataforma y criptomoneda son cosas diferentes y como tales, no siempre se encuentran asociadas.

En definitiva, los mecanismos de tipo PoW no son más que una competición con unas reglas establecidas en la que los nodos mineros ponen a prueba su capacidad de cálculo para obtener una recompensa. Estas recompensas incentivan el crecimiento de la red, lo que incrementa la seguridad del sistema.

2.1.6 Validación

Cuando un nodo minero encuentra un resultado válido, se lanza una señal al resto de nodos de la red. Los demás nodos deben verificar que la solución propuesta es correcta, esto es, validar el nuevo bloque. De acuerdo con ello, podemos acuñar la siguiente definición:

> *"El proceso de **validación** es el mecanismo por el cual un número representativo de nodos acredita que el resultado obtenido en la prueba de trabajo por el nodo ganador es correcto."*

Resolver la prueba de trabajo es, como ya se ha dicho, un proceso costoso. No obstante, si bien encontrar el *nonce* que proporciona un *hash* de características determinadas es una tarea ardua, el proceso inverso es sencillo. Los nodos pueden comprobar fácilmente que el resultado calculado por el nodo que mina el bloque es correcto, y de esta forma se produce la validación del nuevo bloque.

2.1.7 Otros mecanismos de consenso

Otros mecanismos de consenso eliminan la competición del proceso y seleccionan el nodo que ha de minar el bloque por un procedimiento alternativo. Este cambio tiene sentido desde el punto de vista de hacer el proceso más eficiente; hay que tener en cuenta que en los PoW se consume una gran cantidad de energía sólo para mantener la integridad de la red (miles de ordenadores trabajando simultáneamente sobre el mismo algoritmo). Aunque cada plataforma incorpora en sus mecanismos particularidades que los hacen únicos, para los objetivos que aquí se plantean bastará con considerar, además del PoW, los siguientes tipos:

- *Proof of Stake* (PoS). El minado de cada bloque se asigna en función de la cantidad de *tokens* que posee cada participante, típicamente mediante un proceso de selección aleatoria.
- *Proof of Authority* (PoA). Se utilizan una serie de nodos (validadores o *authorities*) cuya función específica es crear los nuevos bloques de la red.

Es interesante destacar el hecho de que, en el caso de los mecanismos de consenso de tipo PoA, el algoritmo deposita su confianza en un grupo reducido de nodos para crear los nuevos bloques. En consecuencia, las redes que utilizan este mecanismo son por definición permisionadas. Por otro lado, es inhabitual que una red privada utilice algoritmos propios de redes públicas; en entornos donde se presupone una cierta confianza, utilizar algoritmos tan costosos en términos energéticos como el PoW se antoja innecesario. Así, el mecanismo de seguridad en estos casos pasa por la capa de privacidad que se aplica a la red.

2.1.8 *Smart Contracts*

El término *smart contract* o contrato inteligente fue utilizado por primera vez en 1994 por el criptógrafo Nick Szabo [11]. Sin infraestructura tecnológica que lo respaldase, el concepto quedó huérfano hasta que la irrupción de Blockchain sacó de nuevo la idea a la palestra.

Como se define en el mismo artículo,

> *"Los* **smart contracts** *son scripts modulares, repetibles y autónomos que se almacenan y ejecutan en una plataforma Blockchain y representan promesas unilaterales de proporcionar una tarea informática determinada."*

Dicho de otra forma, la información registrada en Blockchain puede almacenar cualquier tipo de lógica basada en datos, lo que permite un elevado nivel de automatización.

Los *smart contracts* son, por tanto, piezas de código con dos atributos muy particulares: primero, que son almacenables en la Blockchain y contienen el estado actual del sistema; segundo, que requieren del consenso de la red para ejecutar cualquier cambio.

2.1.9 Plataformas más destacadas

Blockchain	Privacidad	Mecanismo de consenso	Criptodivisa asociada	Capitalización[6] [12] (Millones de dólares)
Bitcoin	Pública	PoW	Bitcoin (BTC)	80.122,83
Ethereum	Pública	PoW	Ether (ETH)	28.257,94
Hyperledger	Privada	PoA[7]	-	-
Ripple	Privada	PoA	XRP	9.814,55
Dogecoin	Pública	PoW	Dogecoin (DOGE)	112,24
Monero	Pública	PoW	Monero (XMR)	1.323,76
Dash	Pública	PoW	Dash (DASH)	2.171,40
Zcash	Pública	PoW	Zcash (ZEP)	545,78
Litecoin	Pública	PoW	Litecoin (LTC)	2.666,76
Nxtcoin	Pública	PoS	Nxt (NXT)	66,98

Tabla 2.1. Clasificación de las plataformas Blockchain más importantes actualmente.

La Tabla 2.1 recoge las plataformas Blockchain más importantes actualmente (todas ellas *open source*). Como se puede observar y en concordancia con lo expuesto en el punto anterior, el PoW es el mecanismo de consenso más ampliamente adoptado por las cadenas públicas, mientras que las privadas optan por algoritmos tipo PoA. No obstante, desarrolladores de importantes plataformas públicas como Ethereum se encuentran trabajando activamente para cambiar de rumbo hacia otros mecanismos en busca de eficiencia.

2.2 Una tecnología más allá de Bitcoin

Los conceptos Blockchain y descentralización se encuentran fuertemente vinculados. Como agente descentralizador, la tecnología Blockchain puede desempeñar una importante labor en cualquier sistema distribuido. Uno de los ejemplos más claros son las denominadas DAO (*Decentralized Autonomous Organization*), organizaciones online basadas en código fuente autónomo que se ejecuta sobre la Blockchain. Todas las funciones de organización están pre-programadas en el código [13]. Este tipo de organizaciones se postulan como un duro competidor para las compañías tradicionales: al grado de automatización implícito en los procesos se suman las ventajas financieras de la gran reducción de los costes generales y la

[6] A fecha de 10 de Octubre de 2017.
[7] Dado el carácter modular de Hyperledger, el diseño del software contempla la posibilidad de utilizar diferentes algoritmos de consenso para que el resultado final se ajuste mejor a la industria y región objetivos. El PBFT (*Practical Byzantine Fault Tolerance*) es el protocolo que implementa en su versión inicial [48]. En este caso se ha creído más conveniente utilizar la denominación genérica de PoA.

carencia de propiedades inmuebles. Colony [14] aspira a ser una de las primeras. Su propuesta es desarrollar una plataforma modular que permita crear equipos de trabajo descentralizados. La plataforma incentiva el buen trabajo premiando a los mejores colaboradores con *tokens* y dándoles una reputación, en base a los cuales pueden reclamar la proporción equivalente de las ganancias de la comunidad.

En el sector financiero, Blockchain se presenta como una herramienta que algunos consideran puede ser clave para operar con mayor transparencia y flexibilidad, así como para reducir los costes [15]. El banco suizo Falcon Private Bank constituye un ejemplo paradigmático de este hecho, al haberse convertido en el primero en su país en ofrecer soluciones de gestión de activos mediante Bitcoin y otras criptomonedas [16]. Tampoco sería justo hablar de Blockchain en el sector financiero sin mencionar Corda [17], la plataforma *open-source* de la *fintech* R3 que cuenta con el respaldo de más de setenta instituciones financieras de todo el mundo [18]. Se trata de una red privada que aspira a convertirse en el estándar de este sector.

Otro ámbito que está suscitando gran interés y que está estrechamente relacionado con la llegada de la denominada industria 4.0 es el del Internet de las cosas (IoT, *Internet of Things*). La empresa alemana Slock.it ha sido una de las pioneras en apostar por un modelo de negocio que combina ambos conceptos. Su principal desarrollo consiste en una plataforma basada en Ethereum que permite el alquiler de activos mediante *smart lockers*. Para alquilar el servicio en cuestión, el usuario simplemente tiene que enviar dinero a la cuenta vinculada al *smart locker*, de modo que se automatiza el proceso. La idea pretende abarcar todos aquellos bienes infrautilizados susceptibles de ser compartidos (apartamentos vacacionales, maquinaria, vehículos, etc) [19]. También es interesante el caso de Filament, una *start-up* que ha desarrollado una plataforma IoT basada en Bitcoin que permite asignar a los elementos de la red eléctrica un número de identificación único [20]. Cada dispositivo conectado puede de este modo enviar datos de su funcionamiento en tiempo real, los cuales se pueden aprovechar por ejemplo para el mantenimiento predictivo de la red.

La atractiva sinergia IoT-Blockchain no sólo ha promovido la proliferación de *start-ups*, sino que grandes empresas consolidadas a nivel mundial, reconocedoras del potencial que encierra, han incluido en sus programas proyectos para el desarrollo de esta tecnología. Tal es el caso de ADEPT (*Autonomous Decentralized Peer-to-Peer Telemetry*), nacido de la

colaboración entre IBM y Samsung y basado en el protocolo de Ethereum. Hasta la fecha el proyecto ha conseguido demostrar con éxito diversas funcionalidades empleando productos reales de Samsung, entre las cuales se encuentran lavadoras que encargan detergente de manera autónoma o gestionan su propio consumo [21]. Otras grandes corporaciones como Airbus o Daimler también están estudiando la implantación de esta tecnología para monitorizar la trazabilidad de sus productos [22].

2.3 Blockchain y el mercado eléctrico

Desde la aparición de Blockchain, la posibilidad de un sistema que permita el intercambio de energía *peer-to-peer* ha sido una idea recurrente. Desde el proyecto Enerchain, del cual se habla más adelante en esta sección, reconocen que cuando se habla de Blockchain y energía típicamente se hace referencia al *trading* P2P, ya sea a nivel microrred o en el contexto de un mercado mayorista. La influencia de la microrred de Brooklyn se ha materializado en la aparición de cerca de una cincuentena de *start-ups* que persiguen esta idea de negocio solo en el continente europeo [23].

Además de las aplicaciones directas en el comercio minorista y mayorista, es posible identificar, al menos, otras tres líneas de desarrollo: la creación de herramientas para la mejora de procesos del mercado, la promoción de la energía renovable y el despliegue de plataformas *open-source* específicamente diseñadas para el sector energético.

2.3.1 Mercados locales y generación distribuida

Es obligado comenzar esta sección por el que efectivamente se erige como el caso paradigmático de la tecnología Blockchain en el sector energético, **la microrred de Brooklyn** [1]. El proyecto está siendo desarrollado por la compañía LO3 Energy, con el respaldo de ConsenSys, e involucra a los participantes pertenecientes a tres redes de distribución del distrito de Brookyln (ver Figura 2.5). La idea es que los vecinos de la zona pueden comerciar con la energía generada localmente, normalmente proveniente de la instalación de paneles solares. La microrred en su sentido más amplio abarca dos estructuras, una virtual y otra física. La capa física es la microrred eléctrica que se construye adicionalmente a la red existente[8] y

[8] La ampliación de la red tiene mucho que ver con la destrucción de la infraestructura que se produjo en 2012 a causa del huracán Sandy. La idea es que la microrred local sea capaz de funcionar de manera independiente (al menos de manera temporal) en caso de una emergencia similar [26].

que tiene por objeto servir como tecnología de *back-up* en previsión de posibles cortes de suministro. Sobre la microrred física se implementa la capa virtual, una plataforma Blockchain privada que utiliza el protocolo Tendermint. Esta plataforma está basada en el desarrollo **TransActive Grid**, fruto de la *joint venture* entre LO3 Energy y ConsenSys [24].

El mecanismo de mercado que se plantea es una subasta simple precio-cantidad que se ejecuta cada intervalo de 15 minutos [1]. En cada intervalo, los *prosumidores* establecen el precio mínimo por el que están dispuestos a vender el excedente de su energía generada y los consumidores el precio máximo al que quieren comprar la energía. El software casa oferta y demanda ordenando a los licitadores de mayor a menor precio. Los consumidores que no son casados, por tanto, recibirían su energía de la red tradicional. No obstante, hay que destacar que, hoy por hoy, la microrred de Brooklyn es alegal; no existe regulación al respecto de este tipo de mercados.

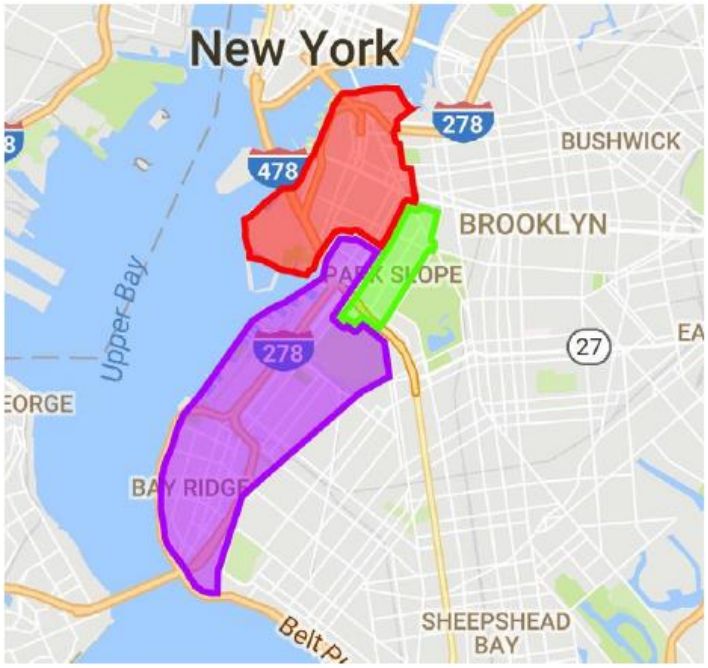


Figura 2.5. Redes de distribución involucradas en la microrred de Brooklyn: Borough Hall (rojo), Park Slope (verde) y Bay Ridge (morado). Fuente: Designing microgrid energy markets. A case study: The Brooklyn Microgrid [1].

Además de colaborar con LO3 Energy para desarrollar TransActive Grid, la compañía ConsenSys (Consensus Systems) está trabajando en un proyecto propio denominado **GridPlus** [3]. A corto plazo, GridPlus pretende operar como compañía comercializadora. Tal y como justifican en su *whitepaper*, su propuesta de valor pasa por ofrecer un precio más bajo a sus clientes que sus competidores, lo cual es posible gracias a la reducción de costes que implica

el nivel de automatización que permite la Blockchain. A largo plazo, la idea es que el cliente disponga de un sistema para optimizar su consumo. Los usuarios podrían almacenar energía para comerciar directamente con otros usuarios o proporcionar servicios complementarios a la red.

Para conseguir los objetivos descritos se requiere del *Smart Agent*, un dispositivo conectado a Internet que permite que el cliente pague por la electricidad consumida en tiempo real. El mecanismo de funcionamiento que se plantea básicamente es el siguiente: el usuario debe en primer lugar comprar y registrar su *Smart Agent*. Después, el cliente ha de pagar a la compañía para que ésta transfiera una cantidad equivalente de BOLT (un *token* desarrollado sobre Ethereum que obedece al estándar ERC-20[9]) al dispositivo del usuario. Desde este momento, el *Smart Agent* puede leer el *smart meter* al que se encuentra vinculado y efectuar los pagos correspondientes en unidades de *token* a través de un contrato que han denominado *"Karabraxos"*. Hoy por hoy el *Smart Agent* de GridPlus es simplemente un prototipo.

También inicialmente orientada al comercio P2P a nivel de comercialización se encuentra el proyecto **Power Ledger**, de la compañía del mismo nombre [25][26]. Desde la empresa quieren desarrollar una plataforma que fomente la evolución del sistema eléctrico hacia un modelo más distribuido. La propuesta de Power Ledger incorpora una novedad interesante: la plataforma involucra a usuarios y compañías eléctricas en un modelo que utiliza dos *tokens*: POWR y Sparkz. El primero se utiliza para permitir acceso a la plataforma, mientras que el segundo es la representación del valor de la energía intercambiada. Las compañías (a las que se denomina *Application Host*) pueden comprar POWR con Ether, Bitcoin o una moneda tradicional, que posteriormente deben transferir a Power Ledger para obtener Sparkz. Las compañías venden entonces Sparkz a los clientes, los cuales pueden utilizar el *token* para comerciar con las propias compañías o con otros *prosumidores*. El mecanismo para incentivar el uso de la plataforma que se plantea es premiar a los consumidores y *prosumidores* que intercambien energía con POWR, que puede ser convertido directamente en Sparkz. Dicho POWR proviene de una pequeña tasa que

[9] ERC-20 es una interfaz estándar que garantiza una buena interacción con otros contratos de la Blockchain Ethereum. Para ajustarse al estándar, el desarrollador debe incorporar una serie de funciones específicas.

Power Ledger obtiene de cada transacción efectuada a través de la plataforma.

2.3.2 De la microrred al mercado mayorista

El potencial de Blockchain en el sector energético va más allá del comercio minorista. A través del denominado proyecto **Enerchain** [27], la compañía germana Ponton señala al menos dos áreas de aplicación adicionales. La primera de ellas se centra en el mercado mayorista, aplicando la tecnología Blockchain al comercio P2P de distintos productos sin necesidad de terceras partes. La primera prueba tuvo lugar en Noviembre de 2016, cuando dos *traders* de las firmas Yuso y Priogen ejecutaron una transacción mediante Enerchain durante el EMART Energy [28], convirtiéndose en la primera en Europa en realizarse a través de Blockchain.

WePower [29] también tiene el punto de mira en el comercio mayorista. En este caso, la idea es facilitar la negociación mediante *"Smart Energy Contracts"*, acuerdos de compra que se ejecutan y registran a través de la red pública de Ethereum.

2.3.3 Blockchain para la optimización de procesos

La otra prometedora propuesta de Ponton busca desarrollar un software basado en Blockchain para la comunicación directa entre agentes del sistema, **Gridchain** [23]. Para el correcto funcionamiento del sistema, la coordinación entre el operador de la red de transporte (TSO, *Transmission System Operator*) y el operador de la red de distribución (DSO, *Distribution System Operator*) es condición *sine qua non*. Procesos de este tipo obviamente se realizan ya hoy en día; sin embargo, Blockchain se presenta como una herramienta de mejora y estandarización. Dado el número de participantes en el mercado, parece interesante sustituir la comunicación bilateral por un sistema que simultáneamente tenga en cuenta todas las partes involucradas, más aún cuando existe un interés creciente por la generación distribuida. Este concepto se aleja por tanto de la idea del comercio P2P y se acerca más a un modelo de negocio B2B (*business-to-business*). Gridchain actualmente se encuentra en estado de prototipo.

2.3.4 Una herramienta para fomentar la energía renovable

SolarCoin [30] introduce una novedad muy interesante: a diferencia del resto de protocolos, en los que las criptomonedas surgen del proceso de minado, en este caso el activo digital se genera al inyectar energía renovable

a la red (en concreto, energía solar). Los productores acreditados reciben una recompensa de 1 SLR (SolarCoin) por cada MWh generado. Así, la comunidad SolarCoin nace de un grupo de voluntarios con el objetivo de incentivar la producción de energía solar.

Otros proyectos de los que se tiene constancia, como Scanergy [31][32][33], están también basados en la misma idea.

2.3.5 Marcos de desarrollo específicos para el sector eléctrico

Fruto de la colaboración entre el Rocky Mountain Institute y la *start-up* Grid Singularity [34] surge **Energy Web Foundation** (EWF) [35], una plataforma Blockchain pensada para servir como infraestructura digital para el desarrollo de soluciones específicas para el sector eléctrico. La idea es crear un marco de desarrollo, respaldado por importantes empresas del sector, que actúe como catalizador de la transición hacia el sistema eléctrico del futuro. Por el momento, se ha desplegado una red privada sobre Ethereum (*"Tobalaba"*).

2.3.6 Blockchain y desarrollo humano

Por último, existen otros proyectos que tienen un carácter eminentemente humanitario. Son proyectos que tienen el punto de mira en países en vías de desarrollo, por lo que la implementación de la tecnología se realiza *bottom-up*, lo cual hace que sean radicalmente distintos a los anteriores. De entre ellos el más destacado es ME SOLshare, una pequeña compañía fundada en Dhaka (Bangladesh) para mejorar las condiciones de vida locales [36].

Grid Singularity, corresponsable del lanzamiento de EWF, también ha participado en proyectos de este tipo. El más destacado consiste en utilizar un *smart meter* programado para aceptar criptomonedas (Bitcoin en concreto) y una plataforma de *crowdfunding* a través de la cual donantes anónimos puedan contribuir para ayudar a zonas empobrecidas [37].

2.3.7 Una clasificación de las iniciativas más destacadas

La Tabla 2.2, incluida al final de este capítulo, resume la información relativa a los proyectos más destacados a fecha de hoy en cuanto a aplicaciones de Blockchain en mercados eléctricos.

Se ha intentado clasificar las iniciativas en función de su objetivo más directo, si bien es difícil delimitar en muchos casos el ámbito de aplicación del proyecto. Además, en cada caso se especifican la zona geográfica donde

se desarrollan, las compañías eléctricas involucradas en caso de que las hubiera, el protocolo Blockchain que utilizan y la criptomoneda o *token* (dependiendo del caso) que emplean, en caso de que exista constancia del desarrollo de un activo digital propio. Al respecto de esto último, es interesante destacar que SolarCoin es el único proyecto de los aquí recogidos que incorpora una criptomoneda propia; naturalmente en el resto de los casos, al no desarrollar una Blockchain de propósito específico, hay que recurrir a una solución alternativa (creación de *tokens*).

La diversidad de objetivos dentro del mismo sector, y el interés que despiertan los proyectos en otras compañías es un indicador claro del potencial de la tecnología Blockchain en el ámbito mercado eléctrico. Sin embargo, la juventud de la tecnología y la escasez de información hacen de Blockchain un vasto territorio inexplorado, dejando patente el hecho de que no están claras las ventajas/inconvenientes frente a las soluciones existentes.

De hecho, es muy significativo que dentro del mismo grupo se planteen diferentes soluciones en cuanto a la plataforma escogida. Por ejemplo, si nos fijamos en el mercado mayorista, WePower utiliza la red pública de Ethereum, mientras que Enerchain ha desarrollado una plataforma de propósito específico basada en el protocolo Tendermint. En el área de aplicación del mercado minorista, por otro lado, el proyecto de la microrred de Brooklyn decidió cambiar el rumbo hacia una solución permisionada tras probar con Ethereum en sus inicios [38]. Todo ello pone de manifiesto que no existe, hoy por hoy, un criterio unánime en cuanto a cómo se puede y debe utilizar la tecnología Blockchain en el campo del mercado eléctrico.

Proyecto	Empresas desarrolladoras	Zona Geográfica	Otras compañías del sector involucradas	Plataforma	Criptomoneda/Token	Aplicación
Brooklyn Microgrid (TransActive Grid)	LO3 Energy, ConsenSys	Brooklyn (Nueva York)	Con Edison	Red privada sobre Tendermint	-	Mercado minorista
Power Ledger	Power Ledger	Australia	Vector Ltd	Ethereum (red pública)	POWR, Sparkz	
GridPlus	ConsenSys	EE. UU.	RWE	Ethereum (red pública)	BOLT	
WePower	WePower	Europa	Novocorex Group, Elering, 220 Energia…	Ethereum (red pública)	WPR	Mercado mayorista
EnerChain	Ponton	Europa	RWE, Enel, Iberdrola, Endesa, Eon, EDP…	Red privada sobre Tendermint	-	
Gridchain						Integración de procesos
Energy Web Foundation	Rocky Mountain Institute, Grid Singularity	Global	Elia, Engie, SP Group, Stedin, TWL, Tepco…	Tobalaba (red privada sobre Ethereum)	Tobalaba Test Tokens	Marco de desarrollo
SolarCoin	SolarCoin Foundation	Global	-	SolarCoin	SLR	Promoción de la energía renovable

Tabla 2.2. Iniciativas más destacadas de Blockchain en el ámbito del mercado eléctrico.

Capítulo 3

Breve introducción al mercado eléctrico

Los mercados mayoristas de electricidad surgen de la necesidad de incrementar los niveles de eficiencia en los servicios de generación a través de la liberalización del sistema.

Durante las décadas de 1980 y 1990, factores de diversa índole propiciaron la transición del modelo monopolista tradicional hacia el nuevo modelo liberalizado en un gran número de países en todo el mundo. Ya en 1970 se detectaron múltiples ineficiencias en el modelo clásico, como señalan diversos autores [39][40]. Este cambio de paradigma ha dado con el desarrollo de mercados organizados de electricidad de diversa tipología.

Todos los países con un sistema eléctrico liberalizado disponen, al menos, de un mercado de corto plazo o *spot* para gestionar el equilibrio entre oferta y demanda, y a través del cual se determina el precio de la electricidad.

Dependiendo de las características concretas de diseño (formato de presentación de ofertas, *pricing*, gestión de servicios complementarios, etc), estos mercados pueden tomar diversas formas, si bien la concepción más extendida está basada en la subasta de precio marginal [40]. Por otro lado, el Operador del Mercado (encargado de gestionar la casación) puede ser independiente o no al Operador del Sistema (encargado de asegurar la viabilidad de los resultados de casación).

Además de mercados al contado, existen mercados a plazo, donde se negocian productos derivados de electricidad y cuyo objetivo primordial es la cobertura de riesgos.

Analizar las características de diseño de los mercados eléctricos está fuera del alcance de este texto; el objetivo de este capítulo es ofrecer una visión general del mercado mayorista, a fin de determinar los procesos en los cuales la tecnología Blockchain puede tener cabida. Para ello, se ha creído conveniente centrar la exposición en el caso más cercano: España. Los contenidos de este capítulo, por tanto, hacen referencia exclusivamente a la operación del Mercado Ibérico de la Electricidad (MIBEL), si bien son muchos de ellos extrapolables a otros mercados.

3.1 Secuencia de mercados

El MIBEL surge de la colaboración entre España y Portugal con el objetivo de integrar los sistemas eléctricos de ambos países [41].

La Figura 3.1 ordena cronológicamente los mercados presentes en el MIBEL. La energía horaria se negocia a través del mercado *spot*, organizado en una sesión diaria, que tiene lugar el día anterior al de entrega, y una serie de mercados intradiarios que tienen lugar el mismo día de la entrega. Esta estructuración en sesiones de contratación sucesivas permite a los agentes ajustarse mejor a sus necesidades en tiempo real. La gestión del mercado *spot* es responsabilidad de OMIE.

Después del mercado diario, al igual que después de los intradiarios, tienen lugar una serie de mercados de restricciones técnicas. Estos mercados están gestionados directamente por el Operador del Sistema y tienen por objetivo asegurar el equilibrio del sistema eléctrico.

Por último, OMIP se encarga de gestionar el mercado a plazo, del cual se hablará extensamente más adelante en este capítulo.

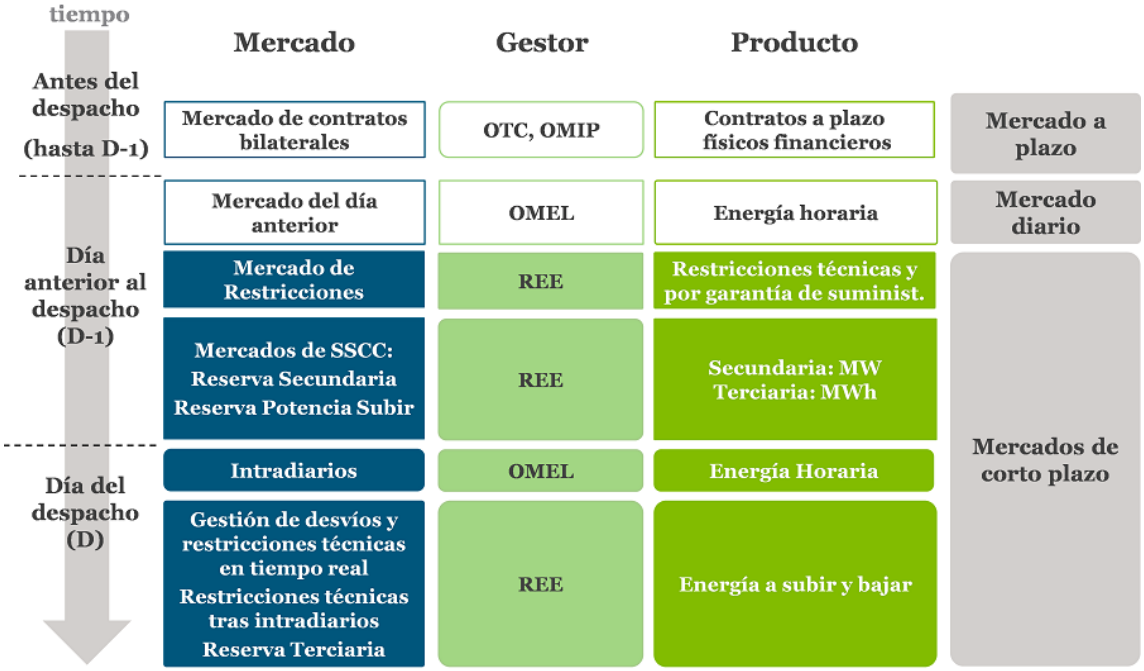


Figura 3.1. Secuencia temporal de mercados en el mercado ibérico de electricidad. Fuente: Manual de la Energía. El mercado mayorista [39].

3.2 El mercado *spot*

El objetivo del mercado al contado, referido habitualmente como mercado *spot*, es establecer el programa de producción y compra de electricidad para el día siguiente al de negociación. Por tanto, el mercado *spot* de electricidad es más estrictamente un mercado del día de antes (*day-ahead*), en tanto que la entrega se produce un día después del cierre de la negociación.

Como se indica en [40], el mercado *spot* es el núcleo de todo sistema eléctrico liberalizado. En España, el mercado al contado cuenta con una componente de contratación diaria y otra de ajustes intradiarios [42].

3.2.1 El mercado diario

El mecanismo del mercado diario permite a los diferentes agentes del mercado presentar sus ofertas de venta y adquisición de energía, y sirve para determinar el precio de la electricidad correspondiente a cada hora del día siguiente. Se trata de una subasta que tiene lugar todos los días del año, de modo que a las 12.00 del día *D*-1, cierre del mercado diario, quedan fijados los precios de referencia para las veinticuatro horas del día *D*.

De manera simplificada, podemos estructurar el proceso en tres sencillos pasos (ver Figura 3.2):

1) En primer lugar, los agentes del mercado envían sus ofertas de compra y venta para cada hora del día. Estas ofertas pueden ser simples (precio-cantidad) o bien incorporar condiciones complejas, como se verá más adelante.
2) El Operador del Mercado (OMIE en este caso) ordena las ofertas y construye las curvas agregadas de compra y venta.
3) El punto de intersección entre las curvas de compra y venta determina el precio de la electricidad para cada hora, así como las unidades generadoras que van a producir en cada período.

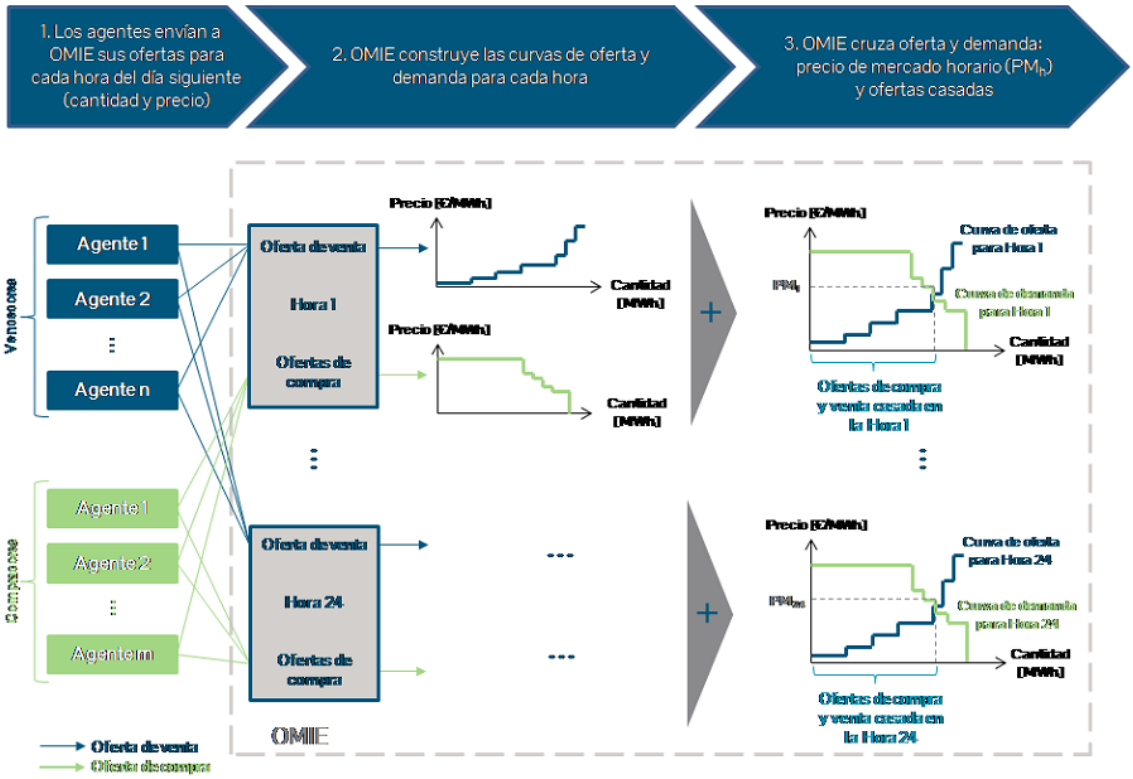


Figura 3.2. Esquema simplificado de procesos en el mercado diario. Fuente: Manual de la Energía. El mercado mayorista [39].

El mercado diario sigue un modelo marginalista, lo que significa que el precio de corte constituye el precio de referencia para todos los compradores y vendedores incluidos en el resultado de casación, independientemente del precio de que figurase en la oferta enviada. Esta casación se realiza a través de un algoritmo denominado EUPHEMIA, de aplicación en todos los mercados europeos.

El algoritmo de casación tiene en cuenta no sólo unidades de oferta simples; las ofertas de venta de energía pueden ser complejas, i.e., incorporar alguna o varias de las siguientes condiciones adicionales:

- Condición de invisibilidad. Permite fijar un valor mínimo de funcionamiento en el primer tramo de cada hora.
- Gradiente de carga. Permite establecer un nivel máximo para la diferencia entre la energía de una hora y la energía de la hora siguiente para la unidad de producción en cuestión. Es una restricción técnica de los generadores que no pueden realizar cambios bruscos.
- Ingresos mínimos. Permite quitar a la unidad de producción del programa, en caso de no alcanzar el ingreso mínimo establecido para el conjunto de horas del día.
- Parada programada. Si la unidad de producción ha sido retirada del programa por no cumplir con la condición anterior, la condición de parada programada le da la posibilidad que realice la parada en un tiempo máximo de tres horas, evitando así pasar directamente a cero. La única condición es que la energía ofertada sea decreciente en cada una de las tres horas mencionadas.

Puesto que el MIBEL integra España y Portugal, los diferentes agentes pueden presentar sus ofertas con independencia del país en que se encuentren hasta que se sature la interconexión. En este último caso, el algoritmo de casación se ejecuta de forma separada, dando lugar a una diferencia de precios entre España y Portugal. Este mecanismo es conocido como *market splitting*.

En consonancia con el párrafo anterior y como se puede deducir, el despacho óptimo desde el punto de vista económico no siempre es viable técnicamente. Las leyes de la electricidad y la topología de la red imponen una serie de restricciones adicionales a la casación del mercado. Para solventar este problema, el Operador del Sistema se encarga de gestionar dichas restricciones, modificando para ello el programa base resultado de la casación. La validación técnica del mercado da lugar al denominado programa diario viable.

Por último, es importante destacar que todas las posiciones abiertas del mercado a plazo en contratos que supongan la entrega física de electricidad se incluyen convenientemente en el mercado diario. Por esta razón, el mercado a plazo de OMIP es el único donde se pueden negociar derivados de electricidad con suministro físico.

3.2.2 El mercado intradiario

El mercado intradiario está estructurado en seis sesiones de contratación (ver Tabla 3.1) que permiten a los compradores y vendedores reajustar sus compromisos hasta cuatro horas antes del tiempo real [43]. Este modelo mediante el cual el agente puede corregir su posición a través de una serie de mercados es referido habitualmente como casación sucesiva [40].

	SESION 1ª	**SESION 2ª**	**SESION 3ª**	**SESION 4ª**	**SESION 5ª**	**SESION 6ª**
Apertura de Sesión	17:00	21:00	01:00	04:00	08:00	12:00
Cierre de Sesión	18:45	21:45	01:45	04:45	08:45	12:45
Casación	19:30	22:30	02:30	05:30	09:30	13:30
Recepción de desagregaciones de programa	19:50	22:50	02:50	05:50	09:50	13:50
Publicación PHF	20:45	23:45	03:45	06:45	10:45	14:45
Horizonte de Programación	27 horas (22-24)	24 horas (1-24)	20 horas (5-24)	17 horas (8-24)	13 horas (12-24)	9 horas (16-24)

Tabla 3.1. Distribución de horarios en las diferentes sesiones del mercado intradiario.

Cada sesión del mercado intradiario funciona de manera similar al mercado diario: se trata de mercados basados en subastas de precio marginal, donde el formato de las ofertas admite condiciones complejas.

3.3 El mercado de derivados

La razón de existencia de los mercados de derivados es dotar a los agentes de un mecanismo para evitar la volatilidad del mercado. Se trata, por tanto, de mercados de cobertura de riesgo de precio. También se denominan mercados a plazo porque la transferencia del activo subyacente del contrato no se efectúa en el momento de negociación, sino en algún momento futuro.

Imaginemos un ejemplo muy sencillo para introducir el concepto. Un generador *A* quiere asegurarse entrar en el programa de generación de todos los días del próximo mes, ya que sufrir una parada le ocasiona costes demasiado elevados. Por otro lado, el consumidor industrial *B* necesita saber que va a disponer de la energía eléctrica necesaria para realizar sus funciones, a un precio razonable. Si *A* y *B* llegan a un acuerdo económico hoy, entonces pueden despreocuparse del comportamiento del mercado durante el mes siguiente, ya que ambos tienen sus necesidades satisfechas.

En el ejemplo anterior, *A* y *B* pueden llegar a un acuerdo por cuenta propia, sin necesidad de un mercado organizado, en lo que se conoce como un contrato bilateral. Sin embargo, existen dos problemas fundamentales:

1) Para que *A* consiga el contrato que persigue, primero tiene que encontrar una contraparte *B* que esté dispuesta a aceptar sus condiciones.
2) En principio, no existen garantías de que, llegado el momento *A* y *B* realicen la transacción que previamente han acordado.

Estas son las carencias que suplen mercados organizados como el de OMIP: por un lado, ejecutan la casación de oferta y demanda, lo que supone un ahorro de tiempo para los negociadores. Por otro, existe una cámara de compensación (OMIClear en este caso) que desempeña el rol de contraparte centralizada de todas las posiciones abiertas, asumiendo el riesgo de impago. Además, el mercado organizado garantiza el anonimato de los participantes, la liquidez y la transparencia.

Lógicamente esta labor no es gratuita, por lo que la participación en este tipo de mercados acarrea un coste más o menos importante para los agentes (ver 3.3.4).

3.3.1 Modelo de mercado

Como ya se ha avanzado antes, existen dos figuras esenciales en el mercado a plazo del MIBEL. OMIP desempeña la labor de operador del mercado, lo que significa que es la entidad responsable de la gestión de la negociación de todas las operaciones. OMIClear, por su parte, asume las funciones de Cámara de Compensación y Contraparte Central en todas las operaciones realizadas en el mercado gestionado por el OMIP, pudiendo también compensar operaciones del mercado OTC o incluso de otros mercados que tengan como activos subyacentes productos de base energética o de naturaleza análoga [44]. Tanto OMIP como OMIClear cuentan con un reglamento propio disponible en la web, de donde se ha extraído la información necesaria para elaborar esta sección [45][46].

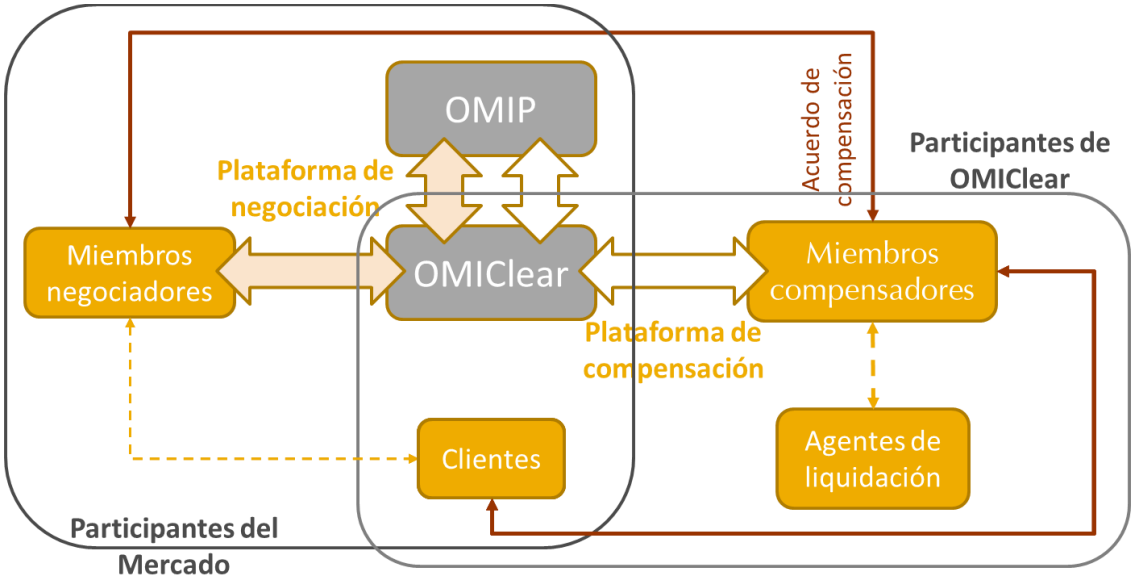


Figura 3.3. Esquema simplificado del modelo de mercado en el mercado a plazo del MIBEL.

Se consideran participantes en el mercado (OMIP) los Miembros Negociadores, los Clientes y los Intermediarios de Operaciones Bilaterales. Los primeros son los participantes que actúan de manera directa en el mercado, pudiendo actuar únicamente por cuenta propia, únicamente por cuenta de terceros (Clientes) o por cuenta propia y de terceros. Los Clientes deben participar en el mercado a través de un Miembro Negociador que tenga autorización para negociar por cuenta de terceros, a no ser suscriba una Operación Bilateral. En dicho caso los Clientes pueden acudir a la negociación a través de un Miembro Compensador.

Se consideran participantes en OMIClear los Miembros Compensadores, los Agentes de Liquidación y los Clientes. Los Miembros Compensadores, que a su vez pueden ser Generales (si tienen facultad para actuar por cuenta propia y de terceros) o Directos (si exclusivamente pueden actuar por cuenta propia), se encargan de compensar las posiciones del mercado, esto es:

1) Registrar las posiciones;
2) Constituir las garantías;
3) Liquidar las posiciones.

Para liquidar las posiciones, es necesario que los Miembros Compensadores figuren en el Sistema de Liquidación financiera; de lo contrario han de celebrar un Acuerdo de Liquidación con un Agente de Liquidación.

Por último, para que un Miembro Negociador pueda acudir al mercado es requisito indispensable que haya firmado un Acuerdo de Compensación con una entidad que posea los derechos de Miembro Compensador.

Todo ello tiene lugar a través de una serie de plataformas específicas diseñadas para tal efecto.

3.3.2 Negociación

La modalidad de negociación de todos los contratos admitidos en el mercado es continua. Sin embargo, OMIP se reserva el derecho de organizar sesiones ejecutadas en subasta, en condiciones y calendario facilitados a los participantes a través de los medios correspondientes.

3.3.3 Productos

Las características más importantes que definen los contratos de derivados negociados en el mercado de OMIP se recogen en la Tabla 3.2.

Producto	Liquidación	Lugar de negociación	PyG	Madurez del contrato
Futuro	Financiera / Física	Mercado	Sí	D, WE, Wk, M, Q, Y
Forward	Financiera / Física	Mercado/OTC	No	Wk, M, Q, Y
Swap	Financiera	OTC	No	D, WE, Wk, M, Q, Y
Opción	Financiera	Mercado/OTC	No	M, Q, Y

Tabla 3.2. Características principales de los diferentes contratos de derivados, según reglamento de OMIP.

Por ejemplo, el contrato de futuros es un contrato a plazo negociado siempre dentro del mercado, por el cual las partes se comprometen a comprar/vender un activo subyacente (energía en este caso, que puede tener liquidación física o exclusivamente financiera) en cantidad y calidad estandarizadas, en fecha y lugar definidos, a un precio acordado en el presente, estando sujeto a liquidación diaria de pérdidas y ganancias en la fase de negociación (PyG).

Si comparamos el contrato de forward con el anterior, podemos comprobar que la diferencia básica es que en este caso el producto no se encuentra sujeto a liquidación diaria de pérdidas y ganancias en negociación. Además, este tipo de contratos se pueden negociar *over the counter* (fuera del mercado).

En la Figura 3.4 se muestran todos los contratos de futuros disponibles actualmente en el mercado de OMIP.

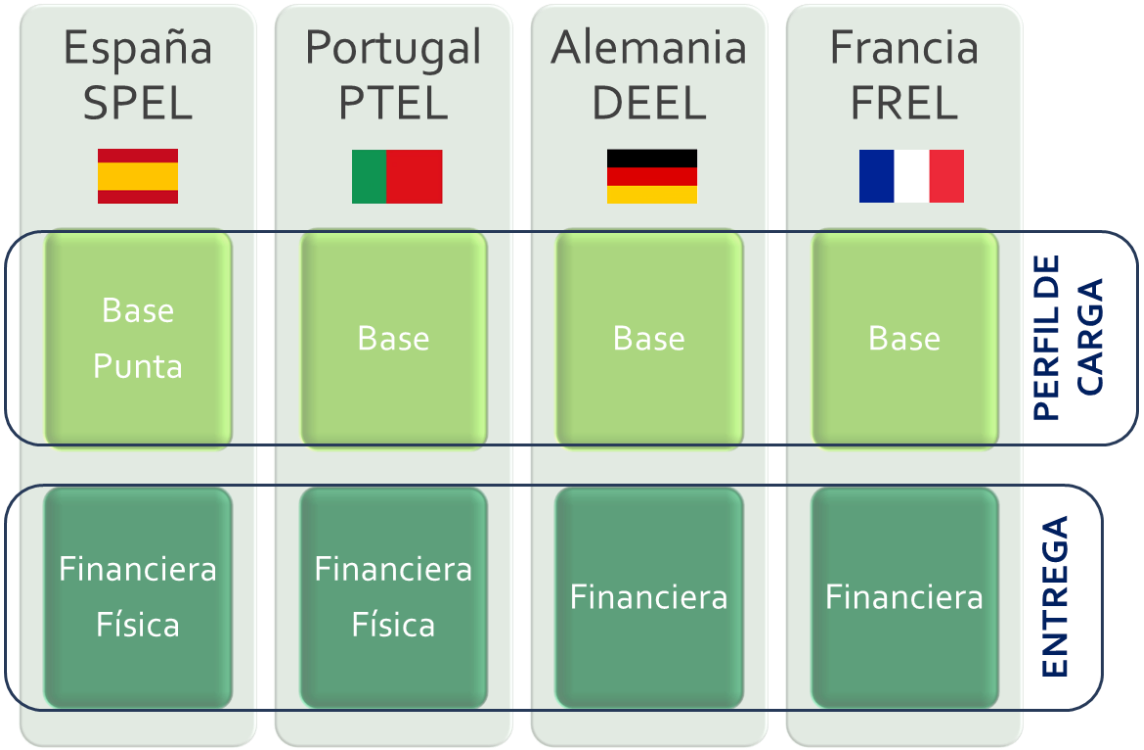


Figura 3.4. Contratos de futuros disponibles actualmente en el mercado de derivados de OMIP.

3.3.4 Costes de transacción

Las tablas que se muestran seguidamente recogen los costes fijos y variables a los que deben hacer frente los miembros participantes del mercado de OMIP y de OMIClear.

Agente	Admisión (€)	Mantenimiento (€)
Miembros de OMIP		
Miembro Negociador (CP/T)	12.000	12.000
Miembro Negociador (CT)	12.000	1.000/cuenta
Miembro Negociador (Light)	1.600	1.600
Intermediario de Operaciones Bilaterales	0	2.000
Miembros de OMIClear		
Miembro Compensador Directo	10.000	10.000
Miembro Compensador General	15.000	15.000

Tabla 3.3. Costes fijos para los agentes de OMIP y OMIClear.

	Comisión (€/MWh)		
Tipo de operación	VM ≤ 1,5 TWh	1,5 TWh < VM ≤ 3 TWh	3 TWh < VM
Miembros de OMIP			
Negociación en continuo	0,0075	0,005	0,0025
Negociación en subastas	0,0075	0,0075	0,0075
Registro de Operaciones Bilaterales	0,0045	0,0045	0,003
Miembros de OMIClear			
Operaciones realizadas en continuo	0,007	0,005	0,0025
Operaciones realizadas en subasta	0,007	0,007	0,007
Operaciones Bilaterales	0,007	0,005	0,0025

Tabla 3.4. Costes variables para los agentes de OMIP y OMIClear.

Capítulo 4

Caso de estudio

Los mercados de electricidad presentados en el capítulo anterior, y por extensión cualquier Exchange, tienen una serie de denominadores comunes: existe un proceso de envío y recepción de ofertas, una casación y, a raíz de ella, una serie de derechos y obligaciones contraídos por los agentes del mercado. En base a ello, podemos encontrar tres grandes grupos de procesos donde la tecnología Blockchain puede dar soporte:

1) Registro de posiciones abiertas por los agentes.
2) Casación del mercado.
3) Generación de contratos entre los agentes, fruto de la casación, y ejecución de los mismos.

Es interesante destacar que el orden de los procesos sólo es estrictamente cronológico en caso de una subasta; en el caso de un mercado continuo (como es el mercado de derivados) no existe una diferenciación clara entre los tres períodos, ocurriendo todos ellos simultáneamente.

Todos estos procesos se resuelven de manera más o menos eficiente hoy en día, por lo que no representan un problema *per se*; sin embargo, es interesante analizar la posibilidad de sustituir los sistemas actuales por una solución Blockchain alternativa. A priori, la casación es el más complejo de los tres computacionalmente, máxime si consideramos la posibilidad de formatos de oferta complejos. Desde la perspectiva de implementación de una red pública, este hecho hace que parezca menos eficiente que el registro de las ofertas y los contratos producto del mercado. No obstante, no hay que perder de vista la posibilidad de utilizar plataformas privadas, donde es posible anular el coste asociado a la ejecución de código.

El objetivo de este capítulo es definir las líneas generales de un caso de estudio que nos permita evaluar el potencial de la cadena de bloques en ámbito del mercado mayorista. Para ello, el capítulo sigue el esquema lógico que se describe a continuación: en primer lugar, se delimita el campo de estudio (4.1). Después, se presenta la arquitectura del sistema considerado, así como las herramientas necesarias para simular dicho sistema (4.2). Finalmente, se incluyen las simplificaciones adoptadas para afrontar el problema, y de acuerdo con ello se propone el modelo conceptual que dará pie a los desarrollos del Capítulo 5 (4.3).

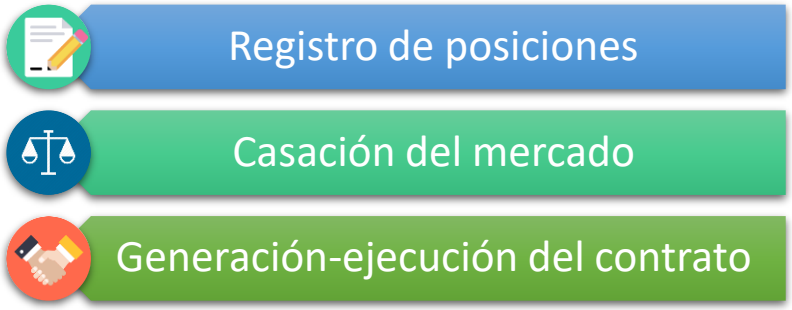


Figura 4.1. Clasificación propuesta de procesos del mercado.

4.1 Marco de aplicación

Dentro del sector eléctrico existen mercados de muy diverso tipo, tal y como se puede colegir del capítulo anterior. Incluso dentro del propio MIBEL se pueden distinguir mercados muy diferentes, tanto en su objetivo como en las reglas que los definen.

En este caso, se ha decidido tomar el mercado de derivados como base para desarrollar el caso de estudio. Este enfoque viene motivado principalmente por dos aspectos:

1) La casación de una subasta como la del mercado diario exige un mayor esfuerzo de cálculo que la casación de un mercado continuo. Para evaluar la capacidad de los sistemas Blockchain en este sentido parece razonable comenzar por el caso más sencillo a nivel computacional. De esta manera, las conclusiones aquí extraídas podrían resultar extrapolables a mercados de tipo subasta: si el sistema falla o da problemas en la casación del mercado de derivados es de esperar que suceda lo mismo con la correspondiente al mercado diario.
2) Muchos de los productos negociados en el mercado de derivados son meros instrumentos financieros. Esto quiere decir que podemos dejar a un lado los productos con entrega física de electricidad y los problemas que ello conlleva, y centrarnos en productos de liquidación exclusivamente financiera. De este modo, es posible implementar en el modelo toda la funcionalidad del mercado, puesto que la liquidación de los contratos correspondientes se traducirá únicamente en el flujo de *tokens* a través de la plataforma.

4.2 Arquitectura

Como ya se ha avanzado en el Capítulo 2, las plataformas Blockchain pueden almacenar y ejecutar lo que se ha denominado *smart contracts* o contratos inteligentes, que en el fondo no son más que piezas de código programable. Cuando nos referimos a la posibilidad integrar los procesos del mercado en Blockchain, lo que queremos, en realidad, es desplegar sobre la red la funcionalidad necesaria (en forma de código) para que dichos procesos se lleven a cabo con el mayor grado de automatización posible[10].

El objetivo, por consiguiente, es construir una estructura basada en *smart contracts* que permita el desarrollo de los procesos identificados. Para ello, es preciso reestructurar dichos procesos en funciones, las cuales a su vez estarán contenidas en los contratos que darán forma al sistema.

[10] Se puede advertir cierta analogía entre el *smart contract* y el concepto de objeto dentro del paradigma de la programación orientada a objetos. En este sentido, las funciones se pueden concebir como métodos del contrato, y sus variables como los atributos que definen su estado.

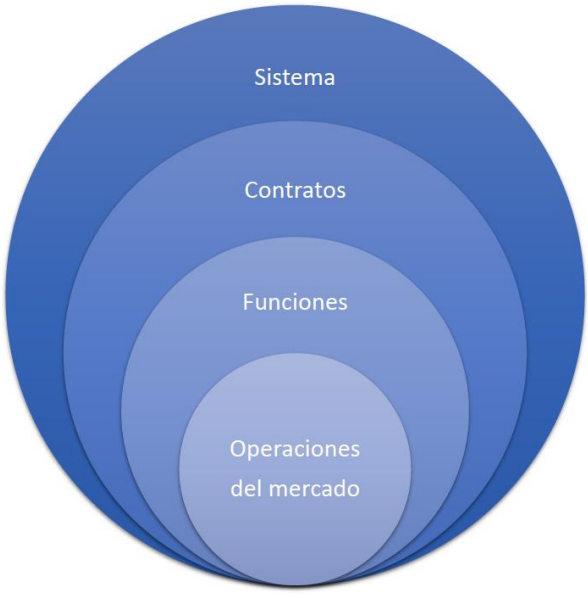


Figura 4.2. Diagrama de Venn apilado de los componentes del sistema.

El siguiente paso es escoger la plataforma sobre la que desplegar el sistema de contratos. En concreto, el caso de estudio se va a realizar en Ethereum por las facilidades que ofrece respecto a otras alternativas. La arquitectura del sistema que se va a simular, por tanto, está conformada por contratos, desplegados sobre la red Ethereum y por consiguiente accesibles para todos los ordenadores conectados a ella, y por los propios ordenadores, a través de los cuales los usuarios pueden interactuar con los contratos (ver Figura 4.3).

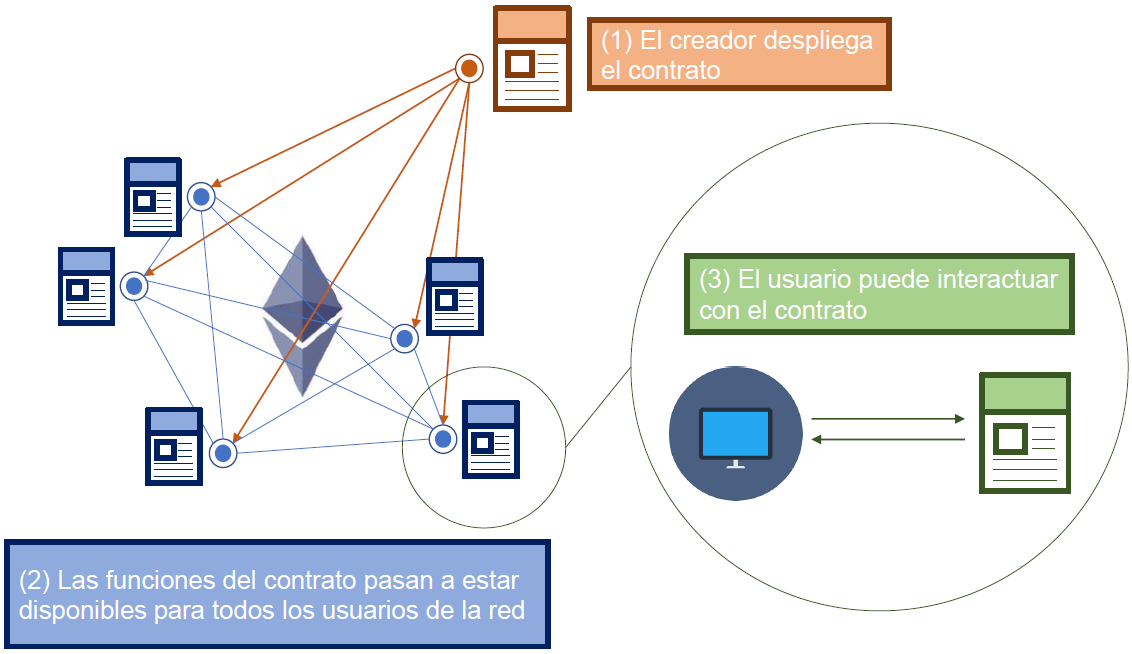


Figura 4.3. Arquitectura del sistema.

Para simular este sistema se ha utilizado Ganache (antes conocido como TestRPC) combinado con Truffle (entorno de desarrollo). La utilización conjunta de estas herramientas proporciona un entorno Blockchain de prueba que permite simular el funcionamiento de la red de Ethereum. De este modo, se pueden desplegar contratos programados en Solidity (lenguaje de programación de Ethereum) y ejecutar las funciones que contienen, utilizando Node.js como entorno de ejecución.

4.3 Modelo conceptual y alcance

A continuación, se presenta el modelo conceptual del sistema considerado (ver Figura 4.4), así como las simplificaciones adoptadas para resolución del problema.

Aunque se utilizan otros contratos, como se verá en el capítulo siguiente, el sistema está estructurado en torno a tres contratos principales:

1) *Token*. Representación del valor única empleada en todo el sistema.
2) Mercado. Representación del sistema de casación.
3) Ticket. Representación de asignación de ofertas casadas.

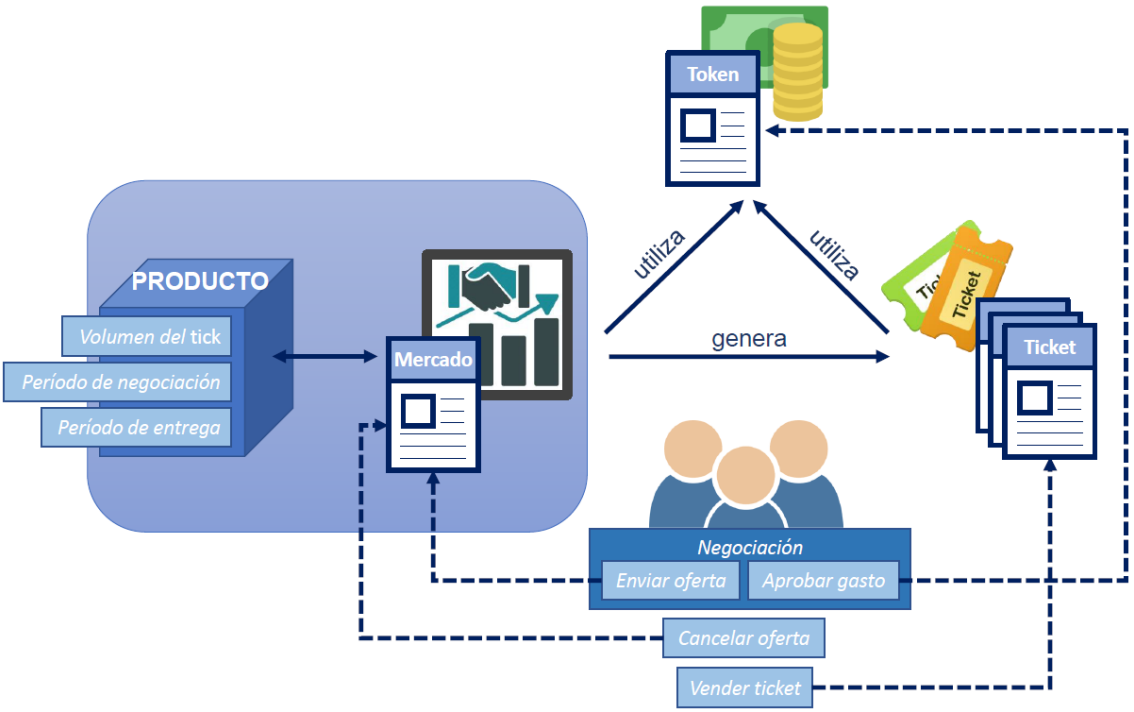


Figura 4.4. Modelo conceptual del sistema.

Como se representa en la figura anterior, a cada mercado le corresponde un único producto y viceversa. Esto implica que cada producto se negocia en un

mercado concreto, i.e., los futuros del día X se negociarán a través de un contrato distinto a los del mes Y.

Puesto que podemos identificar inequívocamente mercado y producto, el mercado quedará definido por las características del producto negociado en él. En concreto y sin pérdida de generalidad, el producto negociado en el mercado será un futuro caracterizado por

i. *un volumen mínimo de negociac*ión (en adelante, volumen del *tick*), que se corresponde con la cantidad (unidades de energía) registrada en cada una de las ofertas, y por extensión en los tickets, como se verá en el capítulo siguiente;
ii. *un período de negociación*, durante el cual es posible enviar ofertas de compra o venta y en el que los tickets generados están sujetos a liquidaciones periódicas de pérdidas y ganancias (*mark to market*);
iii. *un período de liquidación o entrega*, durante el cual no es posible enviar ofertas de compra o venta y se efectúan los movimientos pertinentes para asegurar el cumplimiento del contrato.

La casación se ejecuta en continuo, reproduciendo así la modalidad de negociación del mercado de derivados de OMIP (ver 3.3.2), a través del contrato de mercado. Como se puede observar en la Figura 4.4, para abrir posiciones de compra o venta en el mercado, el agente ha de realizar dos acciones: primero, asignar los derechos necesarios para que el contrato de mercado pueda mover una cantidad suficiente de sus fondos a través del contrato de *token* y segundo, ejecutar el envío de la oferta con los parámetros deseados. Del resultado de casación surgen los contratos que representan la asignación de ofertas casadas, denominados tickets. Dichos tickets, diferentes para comprador y vendedor, incorporan la funcionalidad necesaria para efectuar las liquidaciones derivadas de la responsabilidad adquirida con la compra o venta del futuro, respectivamente.

Para poder transferir el valor entre las cuentas de los participantes, se utiliza el contrato de *token* ya mencionado. Se trata de un contrato adaptado del estándar ERC-20 que incorpora nueva funcionalidad para satisfacer los requisitos que aquí se plantean. Los contratos del ticket y del mercado utilizan las funciones del contrato de *token* para efectuar los movimientos del activo digital necesarios.

Figura 4.5. Secuencia de operaciones del sistema. El contrato de mercado recibe ofertas de compra y venta (1). Con cada nueva oferta, el mercado actualiza sus variables y comprueba el resultado de casación (2); en caso de que éste sea positivo, se despliegan dos contratos de ticket correspondientes a las ofertas de compra y venta casadas (3). El ticket liquida el contrato periódicamente (4).

De esta manera, quedan integrados en el modelo los tres procesos que se pretenden analizar (registro de posiciones, casación y generación-ejecución de los contratos). Las especificaciones inherentes a dichos procesos se analizan en el siguiente capítulo.

Además de acceder públicamente a la información del mercado y participar en la negociación mediante el envío de ofertas, se plantean dos funciones adicionales:

1) cancelar ofertas que hayan enviado y que no hayan sido casadas aún
2) y colocar ofertas para transferir sus tickets, siempre que no haya comenzado el período de entrega.

Por último, es preciso señalar que la gestión de las garantías en el mercado queda fuera del alcance de este TFM. Para los propósitos aquí marcados, bastará con suponer que cada participante dispone en su cuenta de una cantidad de *tokens* suficiente para cumplir con las obligaciones que se deriven del mercado.

Capítulo 5

Prototipo de mercado de derivados

El objetivo de este capítulo es exponer con el máximo grado de detalle posible el prototipo de mercado de derivados desarrollado siguiendo el modelo propuesto en el Capítulo 4. Para ello, este capítulo se centra en analizar sistemáticamente los *smart contracts* del sistema, tanto a nivel individual como su interacción global.

Para no sobrecargar el texto, sólo se han incluido algunos fragmentos del modelo como apoyo a la explicación. Habitualmente dichos fragmentos son cabeceras de funciones o declaraciones de variables, y en algunos casos sentencias relevantes para la comprensión del funcionamiento del sistema. Con el mismo objetivo se incluyen frecuentemente diagramas y figuras que simulan o ejemplifican los subprocesos más importantes.

5.1 Estructura general y consideraciones previas

Como se ha avanzado en el capítulo anterior, el sistema está estructurado en torno a tres contratos principales: un contrato mercado (**Market**), que recibe las ofertas y ejecuta la casación, un contrato ticket, utilizado para representar la asignación de ofertas casadas y realizar las liquidaciones correspondientes al contrato de futuros, y un contrato *token* (**MyToken**) para facilitar el flujo de valor a través de la red.

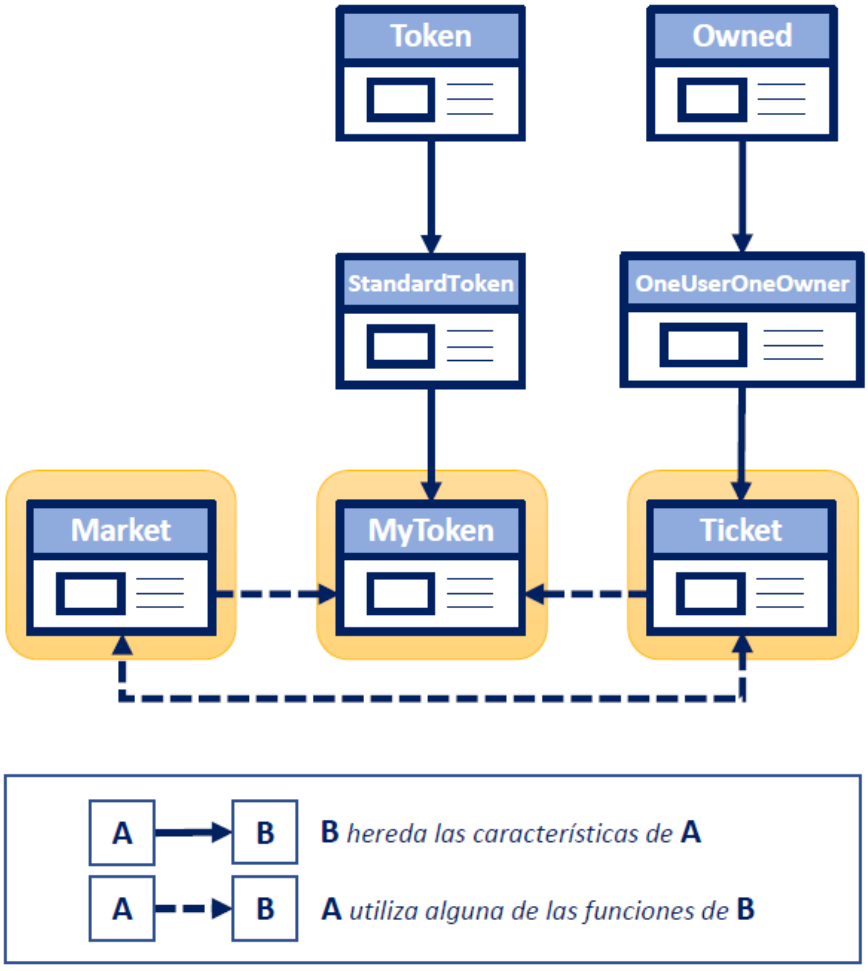


Figura 5.1. Estructura de contratos del prototipo de mercado de derivados.

Además de estos tres contratos, el prototipo de mercado de derivados incorpora una serie de contratos secundarios. Como se puede observar en la Figura 5.1, estos contratos se utilizan básicamente para que los contratos principales adquieran directamente la funcionalidad de los anteriores, aprovechando el concepto de herencia. De este modo, los contratos "hijo" son una ampliación del correspondiente contrato "padre". Esto quiere decir, *verbi gratia*, que el contrato **StandardToken** contará con las mismas

variables y funciones que **Token**, más las incluidas en el propio **StandardToken**. A su vez, **MyToken** heredará la funcionalidad completa de **StandardToken**. Con este sistema se consigue un mayor nivel de modularidad.

Cabe puntualizar que los contratos secundarios no se despliegan sobre la red, motivo por el cual se han omitido en los planteamientos del capítulo anterior; se trata simplemente de una abstracción que nos permite dividir un mismo contrato en varias piezas, de forma que la siguiente siempre contiene la anterior, como si de una muñeca *matrioshka* se tratase.

El concepto de herencia se puede implementar fácilmente en Solidity, tal y como se muestra a continuación a modo de ejemplo:

```
contract StandardToken is Token {…}
```

Por otro lado, en cuanto a la relación entre los contratos principales, destacados en la imagen, es preciso recordar que los contratos **Market** y **Ticket** utilizan el contrato *token* para mover el activo digital de una cuenta a otra. Además, **Ticket** y **Market** se llaman entre sí para ejecutar determinadas acciones. Este hecho es lógico si analizamos separadamente las dos perspectivas:

i. El contrato mercado debe desplegar los contratos ticket, por lo que parece razonable a simple vista almacenar en él una lista con las direcciones de los tickets generados. Esto se puede utilizar, por ejemplo, para que el mercado sepa si una dirección concreta corresponde o no a alguno de los tickets generados.
ii. Desde el punto de vista del contrato ticket, sería ineficiente almacenar en él información que ya está disponible en el contrato mercado. Puesto que el producto negociado es único, vincular ticket y mercado es una manera de acceder a las características del producto sin necesidad de incluir dichas características en el propio ticket.

Además de para acceder a determinadas variables del mercado, almacenar la dirección del mercado en el ticket es útil para desarrollar otras tareas, como se verá más adelante.

Por último, es interesante destacar un aspecto sobre el despliegue de los contratos. Aunque de alguna manera ya se ha avanzado anteriormente en

este texto, conviene tener en cuenta que los contratos **Market** y **MyToken** son desplegados sobre la red por un usuario, al que nos referiremos en adelante como creador. No ocurre lo mismo con el contrato **Ticket**, que no puede ser desplegado por el usuario, sino que se genera desde el contrato mercado como resultado de la casación.

A continuación, se analiza y describe de manera sistemática el funcionamiento de cada uno de los contratos del prototipo de mercado de derivados.

5.2 Representación del valor: *token*

El contrato *token* se utiliza para disponer de una representación única del valor en el mercado, así como para simplificar los procesos de transferencia de valor. Puesto que se trata de un desarrollo basado en ERC-20, el primer paso consiste en analizar este estándar.

5.2.1 El estándar ERC-20

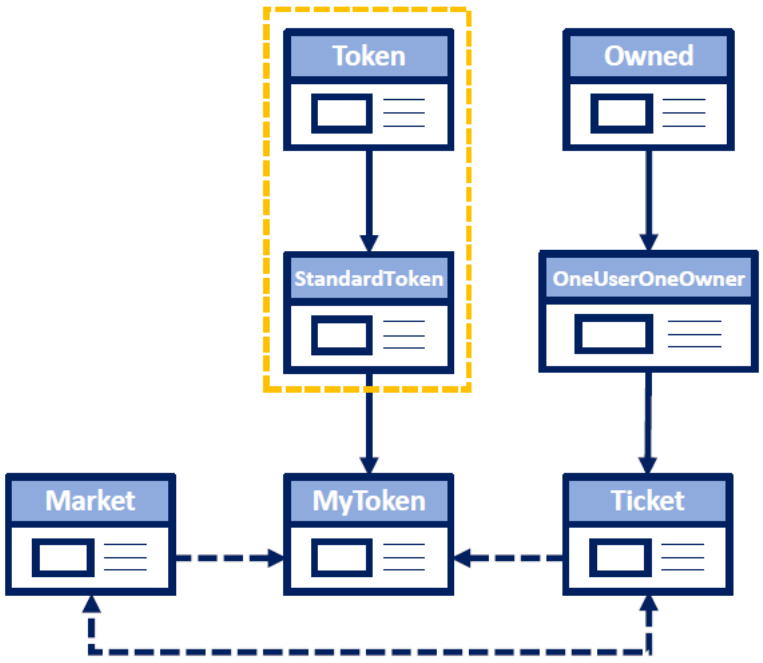


Figura 5.2. Correspondencia del estándar ERC-20 en la estructura de contratos.

ERC-20 es una interfaz estándar *open-source* que garantiza una buena interacción con otros contratos de la Blockchain Ethereum. En el modelo presentado, el contrato ERC-20 se corresponde con los contratos denominados **Token** y **StandardToken**. Se examinan conjuntamente en esta ocasión porque **Token** es simplemente un contrato *abstract*, esto es,

sólo contiene las declaraciones de las funciones que se implementan en **StandardToken**.

Los contratos ERC-20 son relativamente sencillos, y son ampliamente utilizados. Vamos a aprovechar este hecho para introducir algunos conceptos básicos en la programación de *smart contracts*, que serán de utilidad para comprender los desarrollos ulteriores.

Ya es conocido el concepto de *smart contract*; no obstante, no se ha indicado nada aún acerca de la estructura interna del código. Típicamente, un *smart contract* estará formado por los siguientes componentes:

1) variables,
2) funciones y
3) eventos.

Figura 5.3. Componentes de un smart contact.

Las variables, como en cualquier lenguaje de programación, son nombres simbólicos asociados a posiciones de memoria. En un *smart contract*, las variables definen el estado del contrato. En concreto, el estándar ERC-20 dispone de tres variables:

```
(1)  uint256 public totalSupply;
(2)  mapping (address => uint256) balances;
(3)  mapping (address => mapping (address => uint256)) allowed;
```

La variable `totalSupply` (1) es un número entero sin signo que representa la cantidad total de *tokens* en circulación. La utilización de la palabra clave `public` crea automáticamente un *getter* que permite acceder desde otra cuenta a la variable de estado en cuestión; de no ser utilizada, la variable sería privada, esto es, sólo sería accesible desde la instancia particular del contrato a la que pertenece[11], como de hecho ocurre con las otras dos variables.

Tanto `balances` (2) como `allowed` (3) son `mapping`, un tipo de datos más complejo, que de manera simplificada sirve para realizar asignaciones entre valores de distinto tipo. Por ejemplo, en el primer caso lo que se hace es asignar una dirección a un entero sin signo, de modo que `balances[A]` proporciona el balance de la dirección A, es decir, la cantidad de *tokens* que posee.

Vamos a dejar momentáneamente a un lado la variable `allowed` para comenzar a examinar algunas funciones del contrato. Como ya se ha avanzado, la variable `balances` es privada, de modo que no es accesible directamente de manera externa. Para acceder a ella necesitamos una función específica que nos proporcione dicho valor. Dicha función recibe como parámetro la dirección de la que se desea conocer el balance, y devuelve un entero con el valor correspondiente. La sintaxis de la cabecera es la siguiente:

```
function balanceOf(address _owner) public
constant returns (uint256 balance) {…}
```

Por otro lado, el objetivo más básico y primordial de cualquier *token* es permitir la transferencia de valor. Para enviar *tokens* a una cuenta determinada, simplemente hay que llamar a la función `transfer` y pasarle como parámetros la dirección en cuestión y la cantidad que se desea transferir. En este caso, la función devuelve `true` en caso de que la operación se realice con éxito, y `false` en caso contrario.

```
function transfer(address _to, uint256 _value) public
returns (bool success) {…}
```

[11] Este matiz es importante, puesto que dos instancias diferentes del mismo contrato (es decir, el mismo contrato desplegado dos veces dando lugar a distintas direcciones) no pueden acceder a las variables privadas de la instancia contraria.

Lo que llamamos transferencia en el fondo no es más que la actualización de dos variables, en concreto los balances de emisor (A) y receptor (B), como se muestra en el diagrama de flujo de la Figura 5.4. Lógicamente este cambio, como todos los demás, se produce tras su validación por parte de la red a través del mecanismo de consenso.

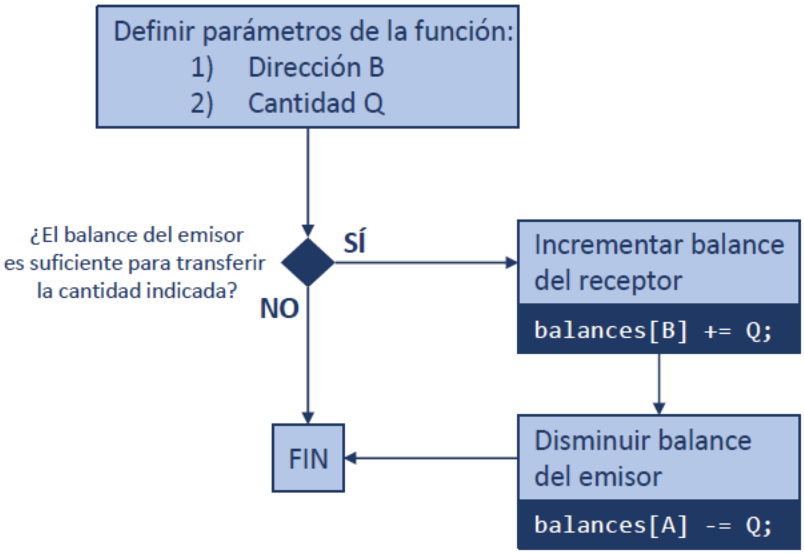


Figura 5.4. Representación de la función "transfer" como diagrama de flujo.

Hasta el momento podemos consultar el balance de una dirección de manera externa y enviar *tokens* desde nuestra cuenta. El contrato ERC-20 va un poco más allá y permite que también podamos mover *tokens* desde una cuenta ajena, siempre que dicha cuenta haya autorizado previamente este gasto. Es en este punto donde entra en juego la variable `allowed`. En este caso se utilizan dos `mapping` consecutivos, de modo que `allowed[A][B]` proporciona la cantidad de *tokens* que A ha aprobado gastar a B, o lo que es lo mismo, la cantidad de *tokens* pertenecientes a la dirección A que B puede mover. Para modificar esta variable y con ello aprobar que otra cuenta utilice nuestros *tokens*, es preciso llamar a la función `approve`.

```
function approve(address _spender, uint256 _value) public
returns (bool success) {…}
```

Una vez se ha aprobado el gasto a una dirección, ésta puede utilizar la función `transferFrom` para hacer efectiva la transferencia. Es importante destacar que no es obligatorio que dicha transferencia se ejecute en una sola llamada; el beneficiario puede utilizar la función `transferFrom` tantas veces

como desee, hasta agotar la cantidad permitida. También es interesante resaltar que la cuenta de destino de la transferencia puede ser cualquiera.

```
function transferFrom(address _from, address _to, uint256 _value) public
returns (bool success) {…}
```

Por último, como `allowed` es también una variable privada, el contrato incorpora un *getter* para dar acceso a su contenido:

```
function allowance(address _owner, address _spender) public
constant returns (uint256 remaining) {…}
```

Analizadas variables y funciones, sólo resta por hablar de los eventos. Los eventos son piezas de información que se despliegan cuando ocurre una acción determinada para notificar del suceso al resto de la red. El ERC-20 implementa por defecto dos eventos: uno para notificar que se ha producido una transferencia, y otro para anunciar que se ha llamado a la función `approve`. A modo de ejemplo se muestra la sintaxis de la declaración del primero:

```
event Transfer(address indexed _from, address indexed _to, uint256 _value);
```

Como apunte, hay que señalar que este evento se lanza de manera idéntica desde `transfer` y `transferFrom`, de modo que dado el evento no es posible saber si procede de la ejecución de una u otra función.

El esquema variables-funciones-eventos utilizado en este apartado será el utilizado de manera general para analizar todos los contratos, si bien en los siguientes no nos detendremos tanto en aspectos técnicos concretos como en su funcionamiento global.

5.2.2 MyToken

MyToken es el contrato *token* que propiamente se despliega sobre la red. Como se genera por herencia a partir de **StandardToken**, **MyToken** cuenta con toda la funcionalidad del estándar ERC-20 analizado en el punto anterior. Además, el contrato cuenta con dos funciones adicionales, las cuales se explican a continuación.

En primer lugar, **MyToken** incorpora un constructor, cuya cabecera se muestra a continuación:

```
function MyToken(
    uint256 _initialAmount,
    string _tokenName,
    uint8 _decimalUnits,
    string _tokenSymbol
    ) public {…}
```

El constructor no deja de ser una función más, pero tiene algunas características que lo hacen un poco más especial. Se trata de una función que comparte nombre con el propio contrato, de manera que cuando se despliega el contrato se llama al constructor. Como cada instancia del mismo contrato se despliega una única vez, el constructor es una función de un solo uso. Llamadas sucesivas al mismo constructor generarán instancias diferentes del mismo contrato.

Además, el constructor es único, lo que quiere decir que no podemos definir distintos constructores para el mismo contrato. Por tanto, la programación de *smart contracts* en Solidity no admite sobrecarga, a diferencia de algunos lenguajes utilizados en programación orientada a objetos como C++.

El uso de un constructor es opcional; podríamos desplegar un contrato sin constructor. En este caso hemos decidido utilizar el constructor para inicializar las variables del contrato. Así, podemos, por ejemplo, personalizar nuestro *token* pasando al constructor como parámetros nombre, símbolo y posiciones decimales. Además, es preciso indicar la cantidad de *tokens* que queremos generar inicialmente.

Como los *tokens* generados se tienen que almacenar en alguna parte, se ha decidido asignar dicha cantidad al creador del contrato, lo cual parece la opción más razonable. En cualquier caso, es trivial programar un reparto distinto. Para realizar esta operación, simplemente tenemos que actualizar el balance de la cuenta del creador de la siguiente forma:

```
balances[msg.sender] = _initialAmount;
```

La palabra clave `msg.sender` devuelve la dirección que ejecuta la llamada al método; en este caso, por tanto, la sentencia devuelve la dirección que

despliega el contrato (creador). Nuevamente, cabe destacar que la generación de *tokens* se traduce simple y llanamente en la actualización de una variable.

Nótese que ni **Token** ni **StandardToken** cuentan con un constructor, puesto que carece de sentido al no desplegarse de manera directa sobre la red.

En segundo lugar, **MyToken** incorpora una función extra de crucial importancia para el funcionamiento del prototipo de mercado. Esta función se denomina `transferAllowance`, y sirve para ceder a un tercero los derechos adquiridos como beneficiario de la función `approve`.

```
function transferAllowance (address _from, address _to, uint _value) public
{…}
```

A continuación, se muestra la representación de la función como diagrama de flujo, considerando A como emisor de la función, B la cuenta de origen y C la cuenta de destino.

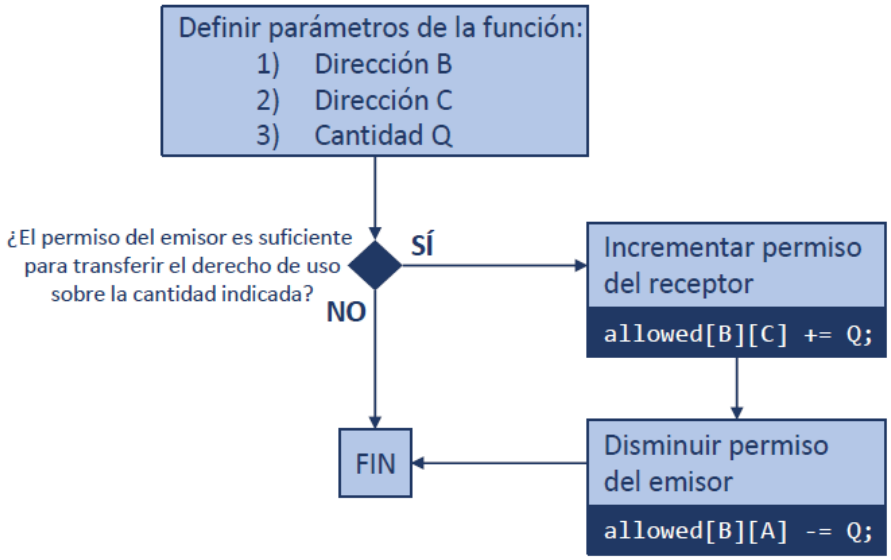


Figura 5.5. Representación de la función "transferAllowance" como diagrama de flujo.

Como se puede comprobar, el funcionamiento es análogo a la función `transfer`, con la salvedad de que en este caso lo que se transfiere no es el activo en sí sino un derecho sobre la utilización del mismo. Aunque no se trata de una función demasiado compleja, vamos a ayudarnos de un ejemplo para mayor claridad.

Supongamos que A aprueba que B utilice una cierta cantidad Q_1 de sus *tokens*. El usuario B no necesita tal cantidad en estos momentos; sin embargo, le gustaría aprobar el gasto de una cantidad Q_2 / $Q_2 < Q_1$ a un tercer usuario, C. Una solución sería ejecutar la función `transferFrom` para transferir los *tokens* desde A hasta su cuenta, para acto seguido aprobar el movimiento de *tokens* por parte de C. Este método se puede realizar con las funciones básicas del estándar ERC-20, pero obliga a que los *tokens* pasen por la cuenta de B y requiere de dos funciones (ver Figura 5.6).

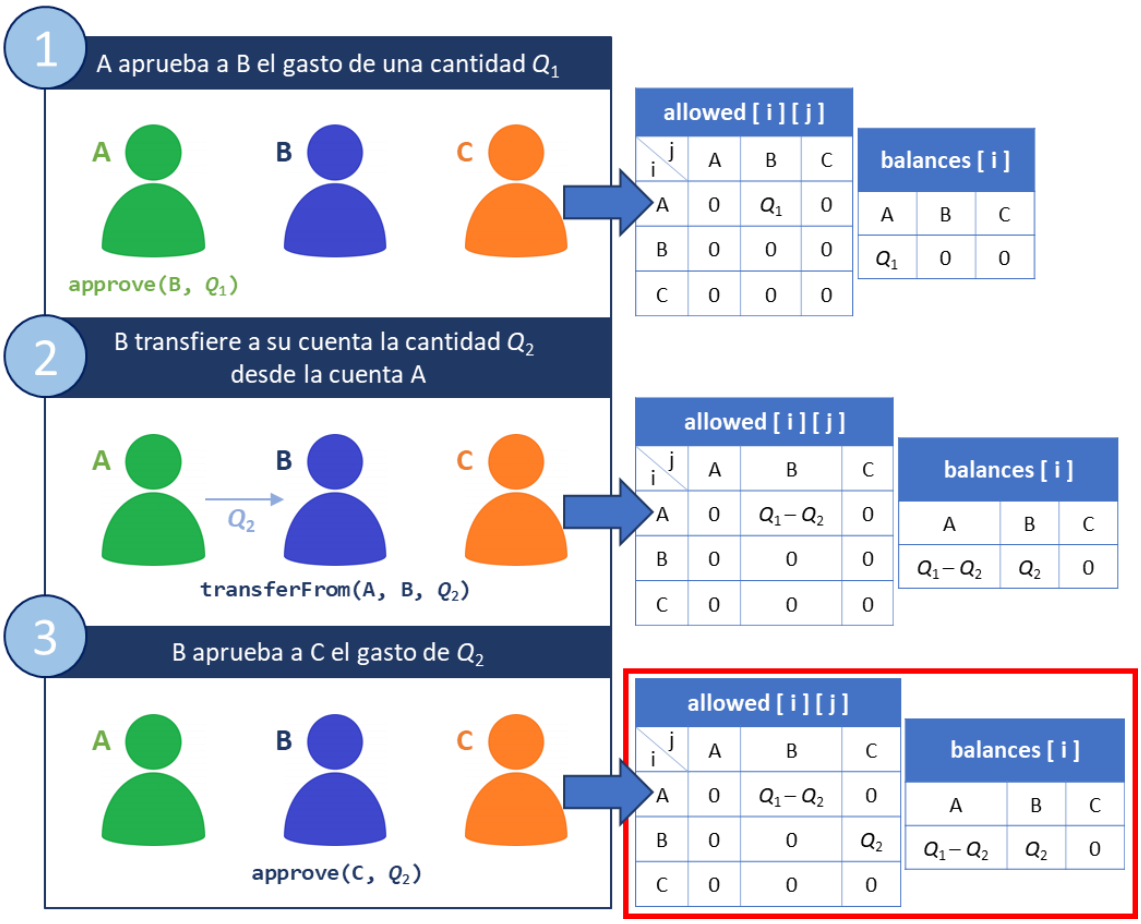


Figura 5.6. Ejemplo de transferencia de permisos, caso A.

La otra opción es que B llame directamente a la función `transferAllowance` para transferir sus derechos sobre los *tokens* de A al usuario C. El resultado en ambos casos es equivalente, tal y como se muestra en la Figura 5.7, pero de esta manera se resuelve el problema en un solo paso, y sin necesidad de que los *tokens* pasen por la cuenta del intermediario. Esta mejora del contrato ERC-20 es especialmente interesante si tenemos en cuenta:

1) la reducción del consumo de gas y por extensión del coste asociado a la transacción;

2) el aumento de velocidad en la ejecución.

Para entender la importancia de todo lo anterior en el caso particular del prototipo de mercado, es necesario analizar la problemática de la liquidación del contrato de futuros, asunto que será tratado en detalle en 5.4.

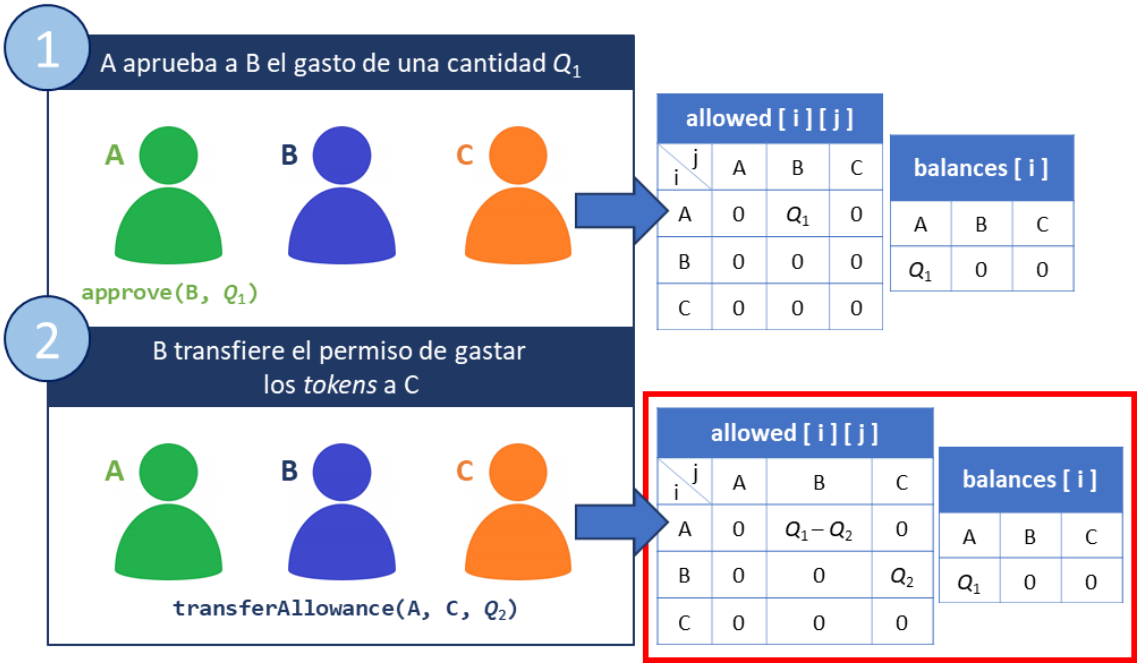


Figura 5.7. Ejemplo de transferencia de permisos, caso B.

5.3 Representación del sistema de casación: mercado

El contrato mercado es el núcleo del prototipo, ya que es la pieza del sistema encargada de casar oferta y demanda. Desde el punto de vista del usuario, son dos las funciones más importantes:

```
(1) function launchOffer (uint _price, uint _quantity, bool _type) public
    {…}
(2) function cancelOffer (uint _offerID) public {…}
```

La primera sirve para enviar ofertas al mercado; la segunda se incluye para permitir que el usuario cancele una oferta que previamente ha enviado, siempre que ésta no haya sido casada. Antes de entrar a explicar dichas funciones, es conveniente presentar las variables más importantes del contrato, puesto que la forma en que está estructurada la información en el contrato es vital para comprender el mecanismo de casación.

5.3.1 Variables

Para optimizar el algoritmo de casación, los precios se ordenan en el vector `prices`, de modo que el valor numérico de un precio concreto se asocia con su posición en el vector. Para establecer la correspondencia entre posición y valor, necesitamos determinar la longitud de dicho vector, N, y una de las siguientes variables:

1) Valor del precio máximo admisible en el mercado, P_{max}.
2) Diferencia entre dos valores de precio consecutivos cualesquiera (escala de precios), s.

Dada una de ellas, la otra se deduce teniendo en cuenta la longitud del vector de precios a través de la relación siguiente:

$$P_{max} = s(N - 1) \qquad (5.1)$$

Aunque la programación del contrato admite cualquier combinación, bastará con considerar un vector de 101 posiciones para el desarrollo del caso de estudio. Simplemente por comodidad, se considera también de manera general que la escala de precios es unitaria, de modo que el precio asociado a prices[k] será directamente k, en unidades de *token*. En definitiva, se dispone de un vector de precios que alberga todos los valores enteros en el intervalo [0, 100]:

```
price [101] public prices;
```

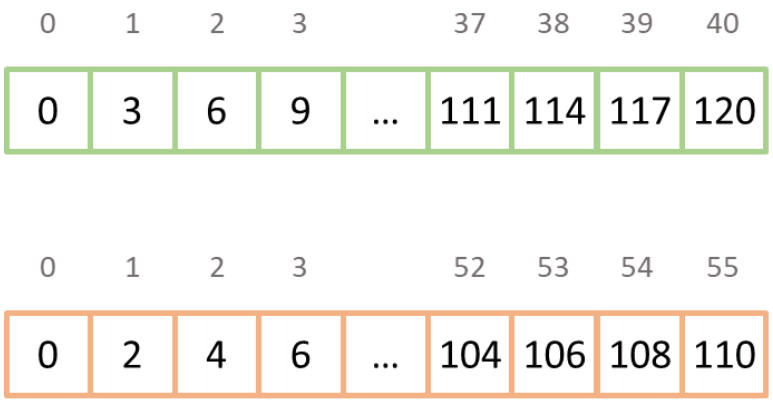


Figura 5.8. Diferentes configuraciones posibles para el vector de precios. Arriba, vector de 41 posiciones y escala 3, lo que permite un precio máximo de 120. Abajo, vector de 56 posiciones, escala 2 y precio máximo de 110.

Modificar el espectro de precios es trivial a partir de la configuración por defecto. Otras posibles configuraciones se muestran en la Figura 5.8.

Si observamos la declaración de la variable `prices`, podemos comprobar que no se corresponde con ninguno de los tipos de datos disponibles en Solidity. El tipo `price` es una estructura definida en el propio contrato:

```
struct price {
    uint value;
    address [] buyingOffers;
    address [] sellingOffers;
    uint buyingAccrued;
    uint sellingAccrued;
}
```

Cuando nos referimos al precio, por tanto, no hacemos referencia exclusivamente al valor pecuniario; en el prototipo de mercado de derivados, el precio es una entidad que contiene cinco variables (ver Tabla 5.1).

Denominación	Tipo	Descripción
`value`	Entero	Valor numérico del precio, medido en *tokens* por unidad de energía
`buyingOffers`	Vector de direcciones	Vector de ofertas de compra local; contiene las ofertas de compra colocadas a este precio, representadas por la dirección del emisor de la oferta
`sellingOffers`	Vector de direcciones	Vector de ofertas de venta local; contiene las ofertas de venta colocadas a este precio, representadas por la dirección del emisor de la oferta
`buyingAccrued`	Entero	Valor acumulado de compra, i.e., número total de ofertas de compra disponibles (no casadas) desde el precio de la oferta de compra más alta hasta este precio
`sellingAccrued`	Entero	Valor acumulado de venta, i.e., número total de ofertas de venta disponibles (no casadas) desde el precio de la oferta de venta más baja hasta este precio

Tabla 5.1. Descripción de las variables contenidas en la estructura "precio".

Los vectores de ofertas de compra y venta locales no son fijos, como ocurre con la variable `prices`, sino dinámicos. Las ofertas se almacenan por orden de llegada en el vector correspondiente, incrementando así su dimensión. Conceptualmente podemos imaginar las ofertas de los vectores locales como bloques que se clasifican y apilan unos encima de otros por orden de llegada, como se muestra en la Figura 5.9. Esta representación de la estructura

`price` es muy útil para entender el funcionamiento del mercado, y será la utilizada en lo sucesivo para ilustrar las explicaciones que lo precisen.

Los valores acumulados, por su parte, son útiles en dos sentidos:

1) Para visualizar virtualmente el comportamiento del mercado.
2) Como herramienta del algoritmo de casación, como se verá en 5.3.2.

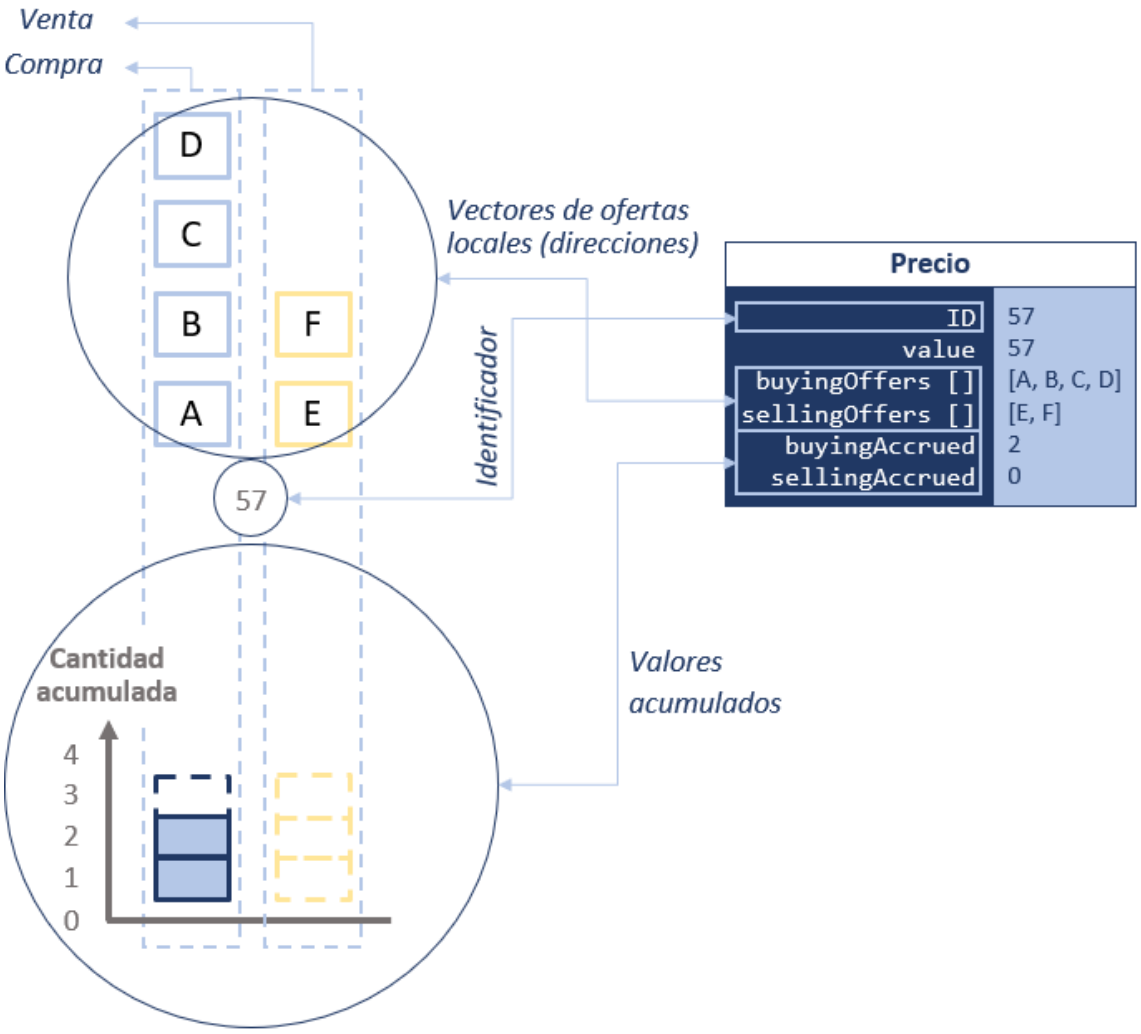


Figura 5.9. Representación de la estructura "precio".

La representación de precio propuesta sirve para reproducir de manera sencilla el estado del mercado. En la Figura 5.10 se ejemplifica una situación aleatoria, considerando por simplicidad un mercado de cinco precios. Nótese que, por la propia definición de las variables, al menos uno de los valores acumulados (compra o venta) será siempre nulo. En el caso anterior, además, se supone que el precio representado es el más alto que contiene ofertas de compra sin casar; de lo contrario, el valor acumulado de compra sería necesariamente mayor.

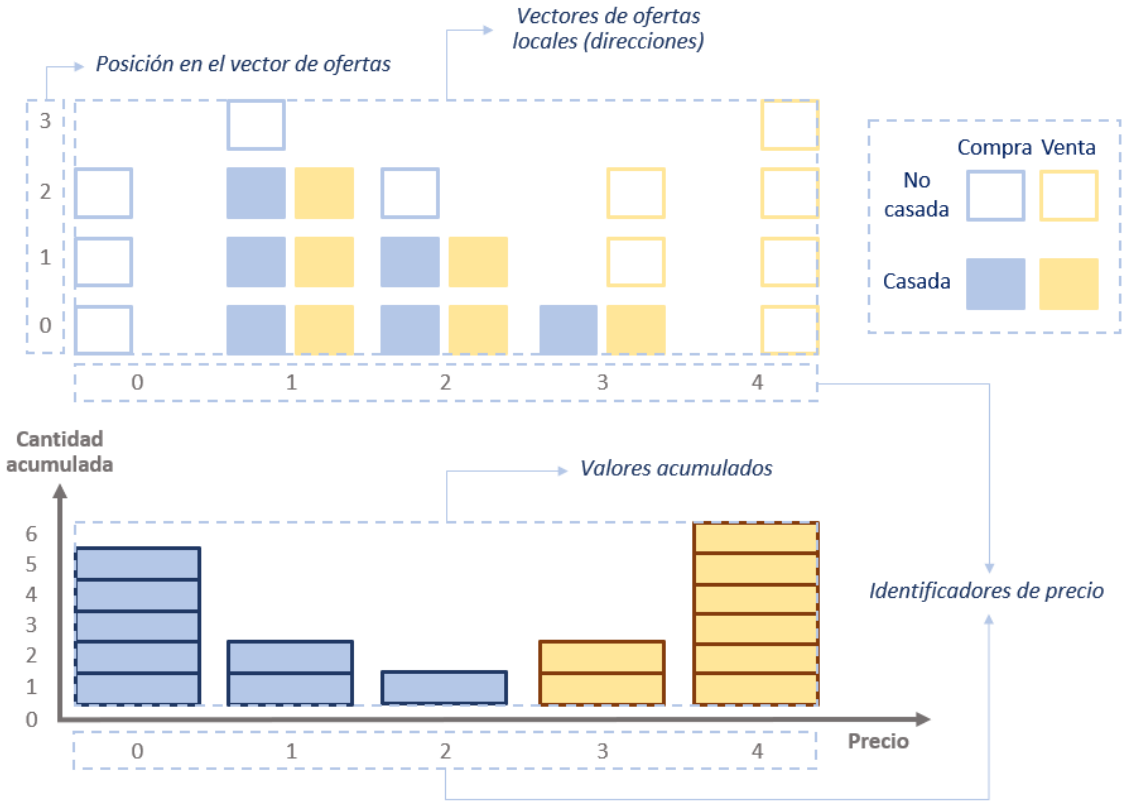


Figura 5.10. Representación de un mercado simplificado de cinco precios a través de las variables del vector de precios.

Todas las ofertas enviadas al mercado se descomponen en ofertas unitarias de volumen igual al mínimo establecido por el mercado (volumen del *tick*). Por este motivo, es suficiente con almacenar en los vectores de ofertas locales la dirección del agente, ya que la cantidad asociada es conocida y de valor igual al volumen del *tick*. En adelante y con carácter general, cuando se haga referencia al término oferta nos estaremos refiriendo en realidad a una oferta unitaria.

De este modo, dado un precio, podemos conocer la lista de ofertas de compra y venta asociadas a él. Sin embargo, necesitamos construir el camino inverso para, dada una oferta, poder acceder a su situación en el mercado. Para ello se hace uso de otro vector de estructuras:

```
offer [] offers;
```

El vector `offers` contiene todas las ofertas del mercado, a diferencia de los vectores de ofertas locales, que únicamente contienen las ofertas indexadas al precio al que pertenecen. Las ofertas enviadas se almacenan de forma ordenada en el mismo instante en que son aceptadas por el contrato,

incrementándose así la dimensión del vector. De este modo, la posición de la oferta en el vector global constituye el identificador único de la misma, análogamente a lo que ocurre con el vector de precios.

Las variables contenidas en el tipo `offer` se muestran en la tabla siguiente:

Denominación	Tipo	Descripción
`priceID`	Entero	Identificador del precio al que se encuentra vinculado, i.e., precio al que ha sido colocada la oferta
`level`	Vector de direcciones	Posición de la oferta en el vector de ofertas local de tipo correspondiente ("oType") del precio al que se encuentra indexada ("priceID")
`oType`	Booleano	Tipo de oferta, i.e., compra (0) o venta (1)

Tabla 5.2. Descripción de las variables contenidas en la estructura "oferta".

La terna de variables se utiliza precisamente para localizar una oferta concreta dentro del mercado, tal y como se muestra en la siguiente imagen:

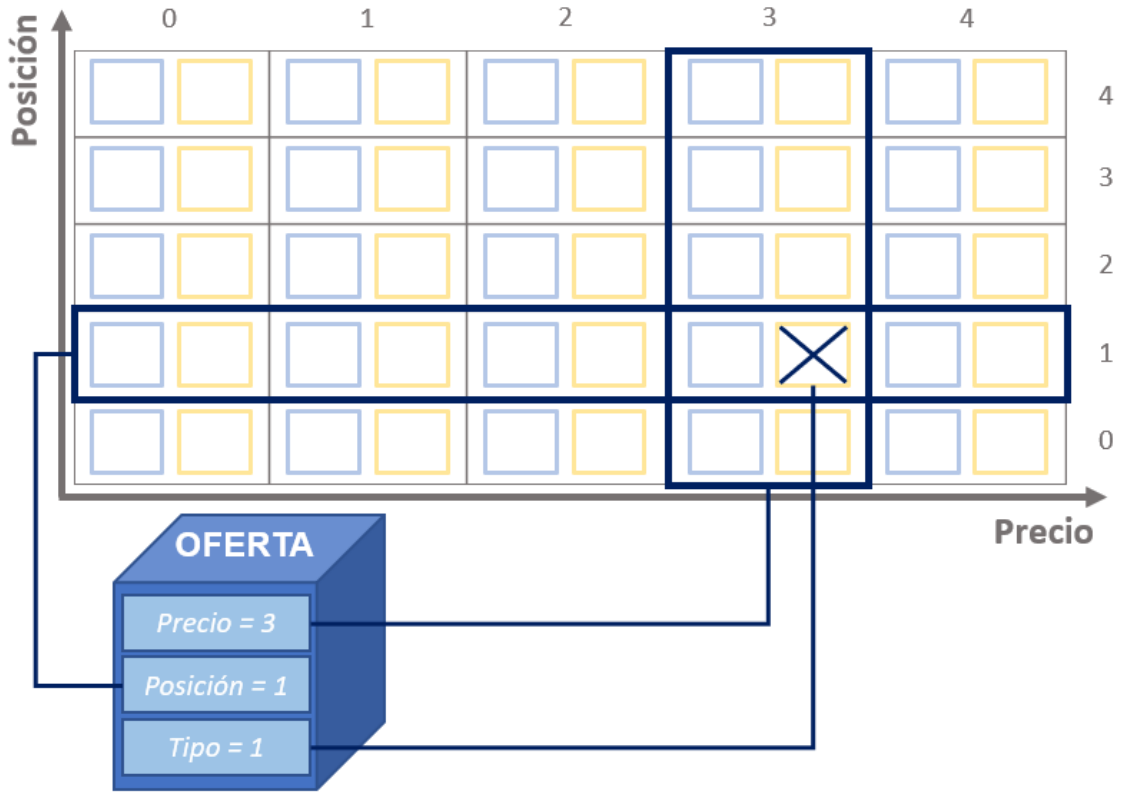


Figura 5.11. Ejemplo de localización de oferta.

Como se puede comprobar, no es suficiente con utilizar las coordenadas de la oferta en el plano precio-posición, ya que en un mismo punto podrían coexistir una oferta de compra y otra de venta. Por este motivo, se hace necesario incluir una variable booleana que indique si nos encontramos en uno u otro caso.

Una vez introducidas las estructuras de datos más relevantes, se pasa a exponer el mecanismo de casación del prototipo de mercado de derivados.

5.3.2 Casación

Para enviar una oferta al mercado, el agente ha de definir tres parámetros, a saber:

1) Precio. Valor al que el agente desea comprar o vender, medido en *tokens* por unidad de energía.
2) Cantidad. Volumen de energía que se desea comprar o vender.
3) Tipo. Valor binario que indica si la oferta es de compra (0) o venta (1).

Nótese que las demás características que determinan el formato de liquidación del futuro (i.e., período de negociación y de entrega) están ya definidas en el contrato de mercado.

La casación se ejecuta en continuo, de modo que con cada nueva llamada a la función `launchOffer` se reproduce el algoritmo de casación, representado como diagrama de flujo en la Figura 5.12.

En primer lugar, es preciso comprobar que el formato de oferta es correcto. En concreto, el algoritmo comprueba que

i. la cantidad introducida sea divisible por el volumen del *tick* del mercado, para que la oferta total se pueda descomponer correctamente en ofertas unitarias;
ii. el precio indicado se encuentra en el rango disponible (por ejemplo, en el caso considerado con 101 valores de precio y escala unitaria, no es posible enviar una oferta con precio superior a 100).

La siguiente comprobación tiene que ver con el contrato de *token*. Si la oferta resulta total o parcialmente casada en algún momento, el contrato mercado ha de tener permiso para mover los *tokens* del emisor a fin de hacer posible la liquidación del producto negociado. Por tanto, el envío de ofertas es más propiamente hablando una secuencia de dos acciones:

1) Asignar derechos suficientes al contrato mercado a través de la función approve del *token* vinculado.
2) Enviar la oferta al mercado mediante `launchOffer`.

Una solución alternativa habría sido imponer una transferencia previa obligatoria del agente al mercado; no obstante, se prefiere utilizar el permiso para no bloquear innecesariamente los *tokens* del usuario en la dirección del

contrato. De esta manera sólo se utilizarán los *tokens* necesarios para satisfacer el contrato de futuros.

Ahora bien, ¿qué permiso es el mínimo necesario para participar en el mercado? Para el presente caso de estudio, se propone un valor a proporcional al precio P y cantidad Q indicados, más un cierto margen:

$$a = (P + 10)Q \qquad (5.2)$$

En cualquier caso, basta con establecer un permiso mínimo lo suficientemente amplio. Es preferible sobredimensionar este valor, en tanto que no entraña riesgo alguno: el mercado sólo puede ejecutar los movimientos para los que ha sido programado.

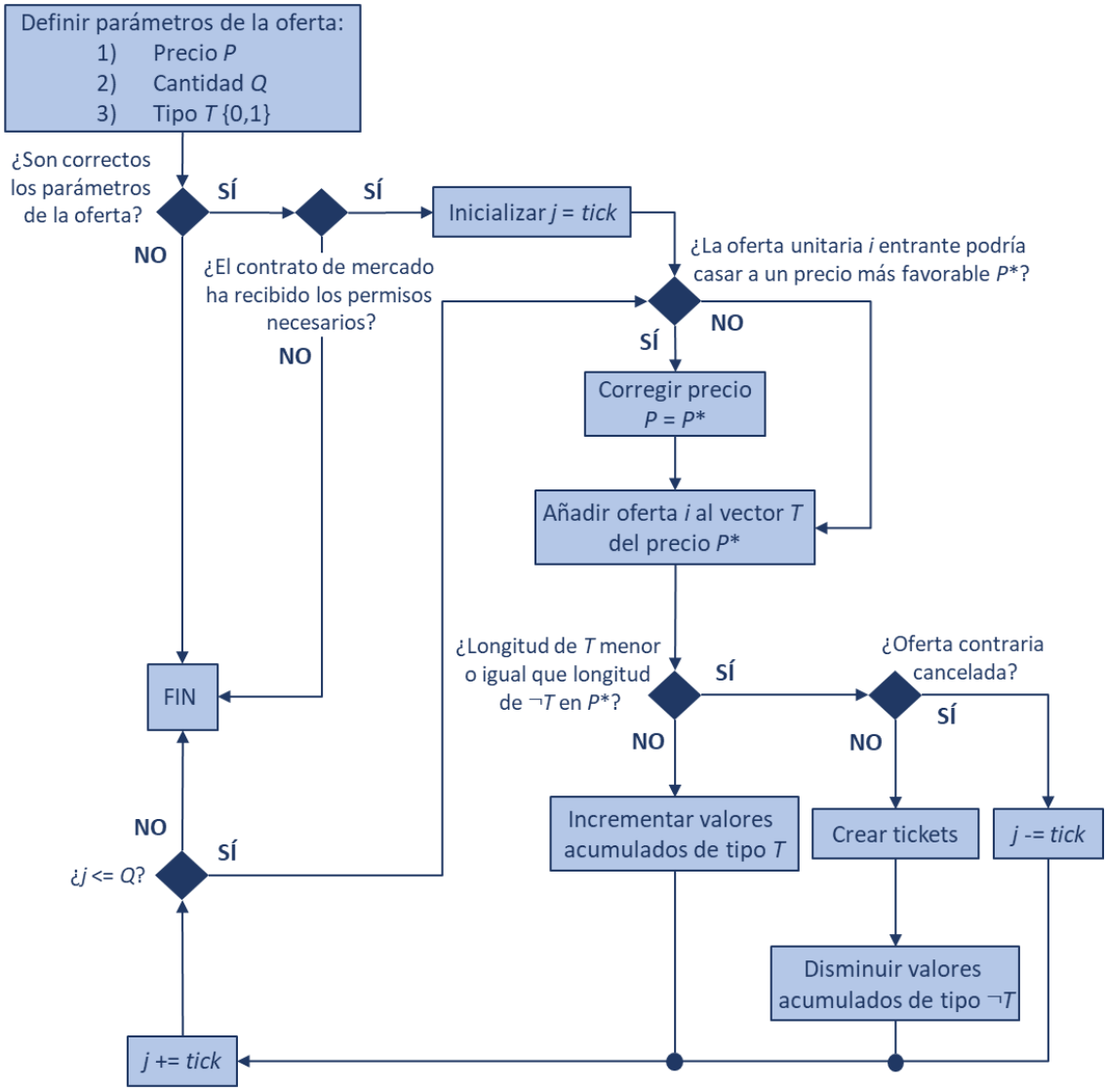


Figura 5.12. Mecanismo de casación del mercado.

Una vez realizadas las comprobaciones pertinentes, el siguiente paso es descomponer la oferta en lo que se ha denominado ofertas unitarias, como ya se ha adelantado en el punto anterior. Puesto que hemos comprobado que la cantidad introducida es múltiplo del volumen del *tick* del mercado, el número de ofertas unitarias n generadas en cada envío será directamente el cociente resultante de dividir la cantidad total Q entre el volumen del *tick*. Para registrar las n ofertas se utiliza el bucle controlado por la variable j. En la primera iteración, j se inicializa al valor del *tick*. En cada vuelta el contador se incrementa en la misma cantidad, de forma que, en la última iteración, j es igual a Q. Puede comprobarse que al finalizar el bucle se han procesado las n ofertas deseadas.

Cada oferta unitaria sigue el siguiente proceso:

1) Se comprueba que el precio indicado P es igual al precio más favorable al que podría casar instantáneamente, P^*. En caso contrario, se corrige el precio de la oferta de forma que $P = P^*$. Es decir, imaginemos que existen ofertas de compra a precios de 45 y 50 *tokens*, y alguien envía una oferta de venta a 45. Aunque casaría directamente a ese precio, para el vendedor es más favorable que la oferta case a 50, y además se evita la posibilidad de hacer el *bypass* a la oferta de compra más meritoria (la de mayor precio en este caso) [12]. Para realizar esta corrección, se aprovechan los valores acumulados de compra y venta registrados en cada estructura precio. En concreto, cuando se envía una oferta de venta (compra), se comprueba si el valor acumulado de compra (venta) del precio siguiente (anterior) es nulo; en caso afirmativo, no existen actualmente ofertas a un precio más favorable, coincidiendo el precio de la oferta con el óptimo. En caso contrario, existe al menos una oferta más favorable con la que realizar la casación.
2) Se coloca la oferta en el precio correspondiente, que en este momento será siempre P^* (ya sea por corrección o no).

[12] Esto es, en cierto modo, una decisión de diseño del mercado. Una alternativa habría sido ordenar las ofertas de manera cronológica, en cuyo caso la oferta más meritoria sería la de mayor antigüedad.

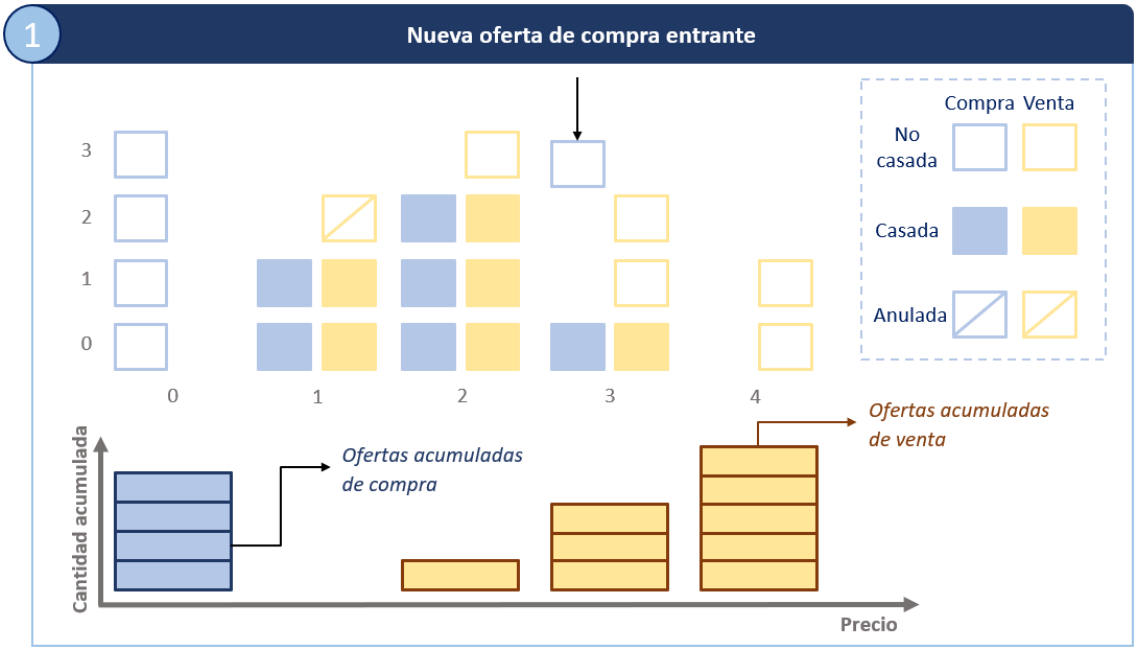


Figura 5.13. Ejemplo de corrección de precio (1).

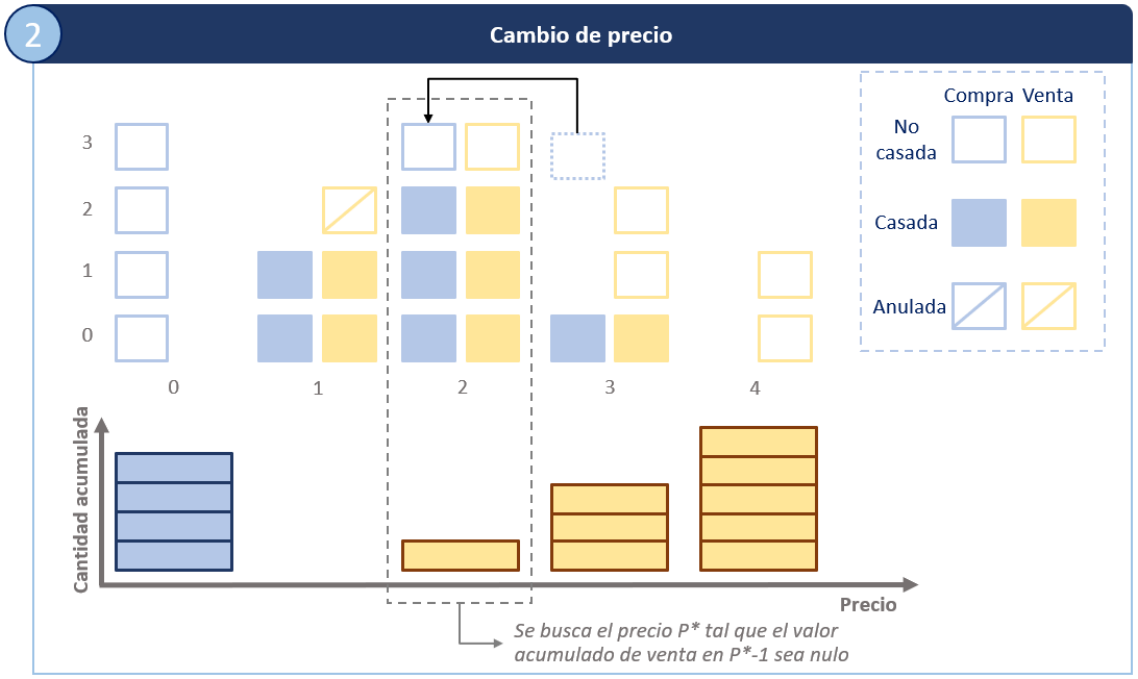


Figura 5.14. Ejemplo de corrección de precio (2).

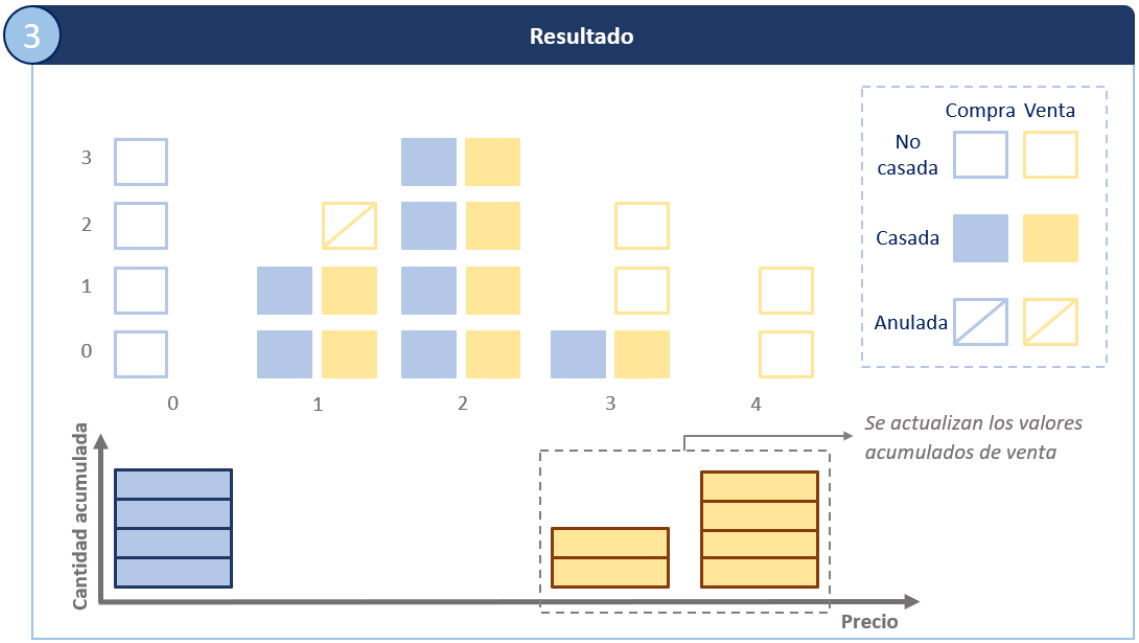


Figura 5.15. Ejemplo de corrección de precio (3).

3) Para comprobar si el resultado de casación es positivo podemos utilizar nuevamente los valores acumulados, pero es más conveniente medir directamente las longitudes de los vectores de ofertas del precio en cuestión. Así, si el vector de ofertas del tipo especificado en el envío, T, es de tamaño menor o igual que el vector de tipo contrario, $\neg T$, entonces la oferta ha casado. Y no sólo eso, sino que la estructura de datos utilizada nos facilita el acceso a la oferta complementaria: la dirección de la contraparte correspondiente se encuentra en la posición N-1 del vector de tipo $\neg T$, siendo N la longitud actual del vector de tipo T. Este sistema nos ahorra la operación de recorrer el vector, optimizando el proceso de casación.
4) Si la oferta casa, el contrato genera los tickets correspondientes, teniendo en cuenta que las direcciones de los agentes beneficiarios son conocidas.
5) Finalmente, tanto si la oferta es casada como si no, se han de actualizar los valores acumulados de compra o venta de manera conveniente.

Existe un caso que no se ha contemplado en la secuencia arriba descrita. Puede ocurrir que, siendo positivo el resultado de la casación, el algoritmo advierta que la oferta complementaria ha sido anulada previamente; en este caso, la solución pasa por disminuir la variable j, como se puede observar en el esquema adjunto. Lo que conseguimos con esto es no contabilizar la oferta actual, puesto que ha quedado inutilizada. Es lo que en adelante

denominaremos oferta *dummy*, una oferta falsa cuya única misión es ocupar la posición complementaria a una oferta cancelada. Nótese, además, que en este caso no se actualizan los valores acumulados.

No obstante, es importante que la oferta *dummy* sea colocada en el mercado, puesto que la asignación de tickets se basa en la longitud de los vectores de ofertas.

Por último, cabe destacar que los valores acumulados no tienen en cuenta aquellas ofertas que han sido canceladas, como se verá en el punto siguiente. Este aspecto es fundamental, ya que impide que el algoritmo lleve a cabo una corrección de precio para generar una oferta *dummy*.

5.3.3 Cancelación de ofertas

En el algoritmo de casación se resuelve el problema de gestión de las ofertas canceladas mediante la generación de ofertas *dummies*. Ahora bien, ¿cómo sabe el sistema si una oferta está cancelada? Para resolver este problema, el contrato mercado utiliza la variable `cancelled`:

```
mapping (uint => mapping (uint => bool)) public cancelled;
```

Esta variable se puede interpretar como una matriz de booleanos en la cual las columnas representan el identificador de precio a y las filas la posición en el vector de ofertas local b, y cuyos valores son:

a) 1, si existe una oferta cancelada en (a, b);
b) 0, en caso contrario.

Se puede apreciar que el concepto es similar al expuesto en 5.3.1 para la localización de ofertas. No obstante, en este caso no necesitamos el tipo de oferta por una explicación muy simple. Imaginemos que se produce la casación de una nueva oferta. En esta situación, es claro que la oferta entrante no está anulada, puesto que la cancelación es siempre posterior al registro de la oferta. Si el valor de `cancelled[p1][p2]` es 1, siendo p_1 el precio de la oferta entrante y p_2 su posición en el vector local, entonces significa necesariamente que la oferta de tipo contrario situada en las coordenadas (p_1, p_2) ha sido cancelada, y la oferta entrante queda automáticamente inutilizada. Esta es la comprobación que el algoritmo ejecuta para saber si la oferta contraria es válida cuando el resultado de casación es favorable.

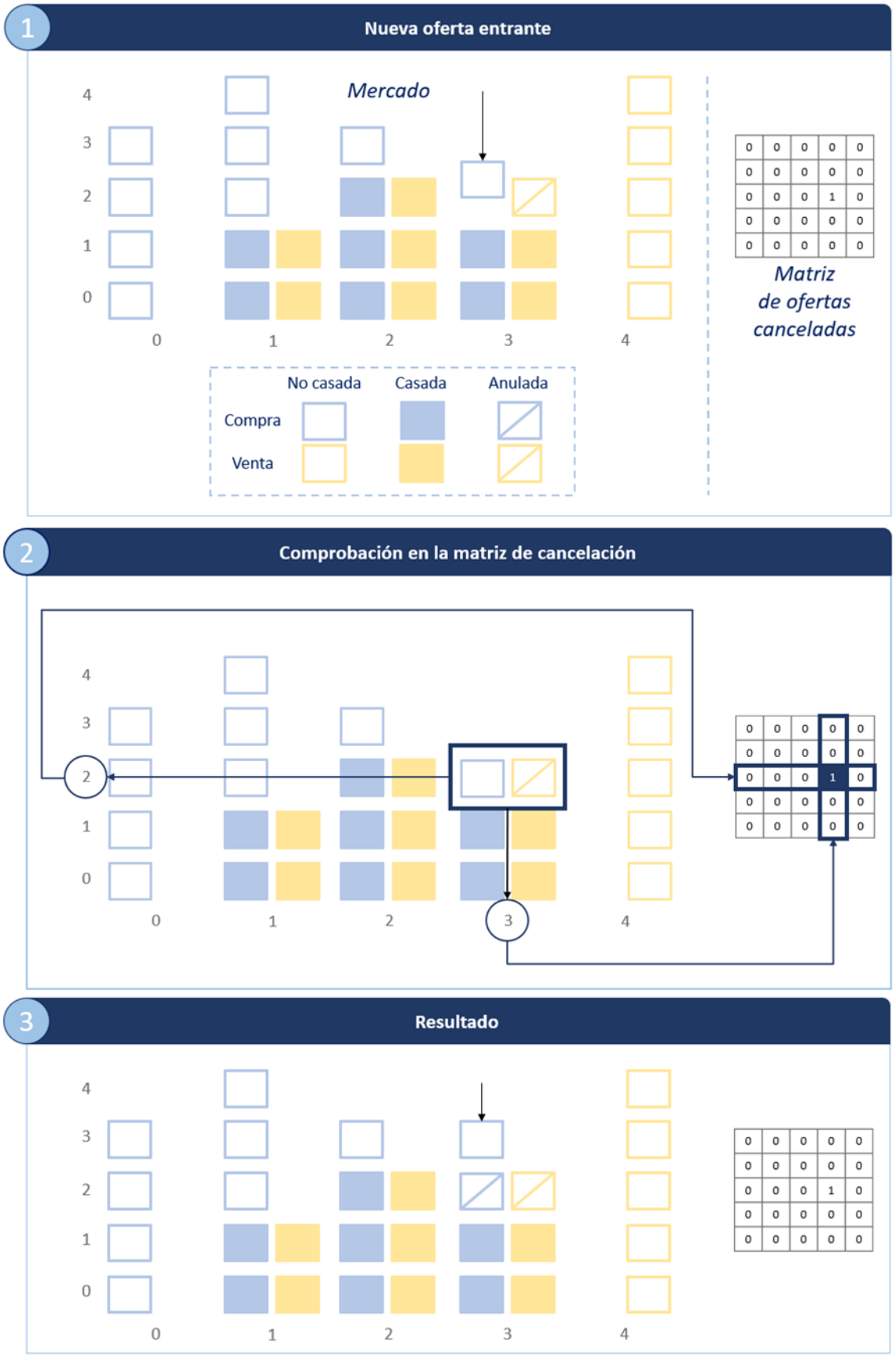


Figura 5.16. Ejemplo de casación con oferta cancelada.

Para cancelar una oferta el usuario simplemente debe llamar a la función correspondiente, escribiendo el identificador de la oferta en cuestión. El sistema localiza la oferta en el mercado a partir del identificador, lo cual proporciona la dirección del emisor de la oferta, y se comprueba que coincide con la dirección del usuario. Lógicamente sería un grave error permitir que un usuario pudiese cancelar ofertas ajenas.

Una vez verificado que la dirección es correcta, se pone a 1 la posición correspondiente de la matriz de ofertas canceladas y se actualizan los valores acumulados para desestimar la oferta cancelada.

No es necesario comprobar que a la oferta que se intenta cancelar no se le haya asignado ningún ticket, ya que se trataría de una cancelación ficticia, al no tener ningún efecto en el mercado. Sí es preciso, en cambio, realizar esta comprobación antes de actualizar los valores acumulados de compra y venta (ver Figura 5.17).

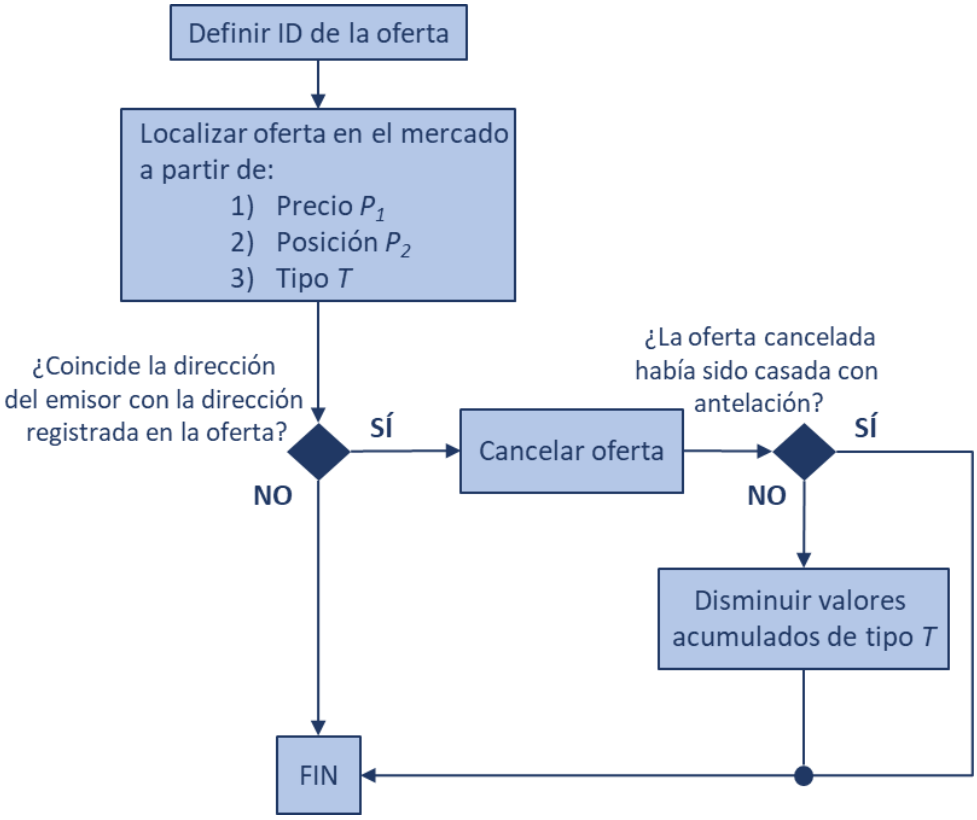


Figura 5.17. Esquema del proceso de cancelación de una oferta.

5.4 Representación de asignación de ofertas casadas: ticket

En el modelo de mercado propuesto, el ticket es la abstracción que representa la asignación de ofertas casadas. Incluye, además, la funcionalidad necesaria para efectuar las liquidaciones derivadas del contrato de futuros.

5.4.1 Atributos y ciclo de vida

Para posibilitar la ejecución de las liquidaciones pertinentes a través de la plataforma, el ticket ha de incluir la siguiente información:

Denominación	Tipo	Descripción
`ID`	Entero	Identificador único del ticket
`mkt`	Dirección	Dirección del mercado que ha generado el ticket
`token`	Dirección	Dirección del contrato token vinculado al mercado
`user`	Dirección	Dirección del propietario del ticket
`ticketPrice`	Entero	Precio resultante de la negociación
`ticketType`	Booleano	Tipo de ticket, i.e., compra (0) o venta (1)

Tabla 5.3. Descripción de los atributos del ticket.

Estos atributos son variables inicializadas en el momento de nacimiento del ticket, utilizando para ello el constructor del contrato:

```
    function Ticket (
        uint _ticketID,
        address _marketAddress,
        address _tokenAddress,
        address _agentAddress,
        uint _price,
        bool _type
        ) public {
```

Los tickets se generan por pares desde el mercado en el mismo instante en que se produce la casación. Como se ha indicado en 5.3.2, en el momento de la casación el contrato mercado deduce las direcciones de comprador y vendedor a partir de la longitud de los vectores correspondientes, de modo que se dispone de toda la información necesaria para desplegar los tickets. Cabe llamar la atención sobre el hecho de que no se incluya ningún atributo relacionado con la cantidad; hay que recordar que los tickets corresponden a ofertas unitarias de volumen conocido e igual al volumen del *tick* del mercado.

La creación de tickets lleva consigo una secuencia de ejecución concreta que va más allá del despliegue de los contratos, y que se ha preferido omitir hasta el momento. Si diseccionamos la caja negra "crear tickets" del flujograma de casación, lo que tenemos es la secuencia de la Figura 5.18.

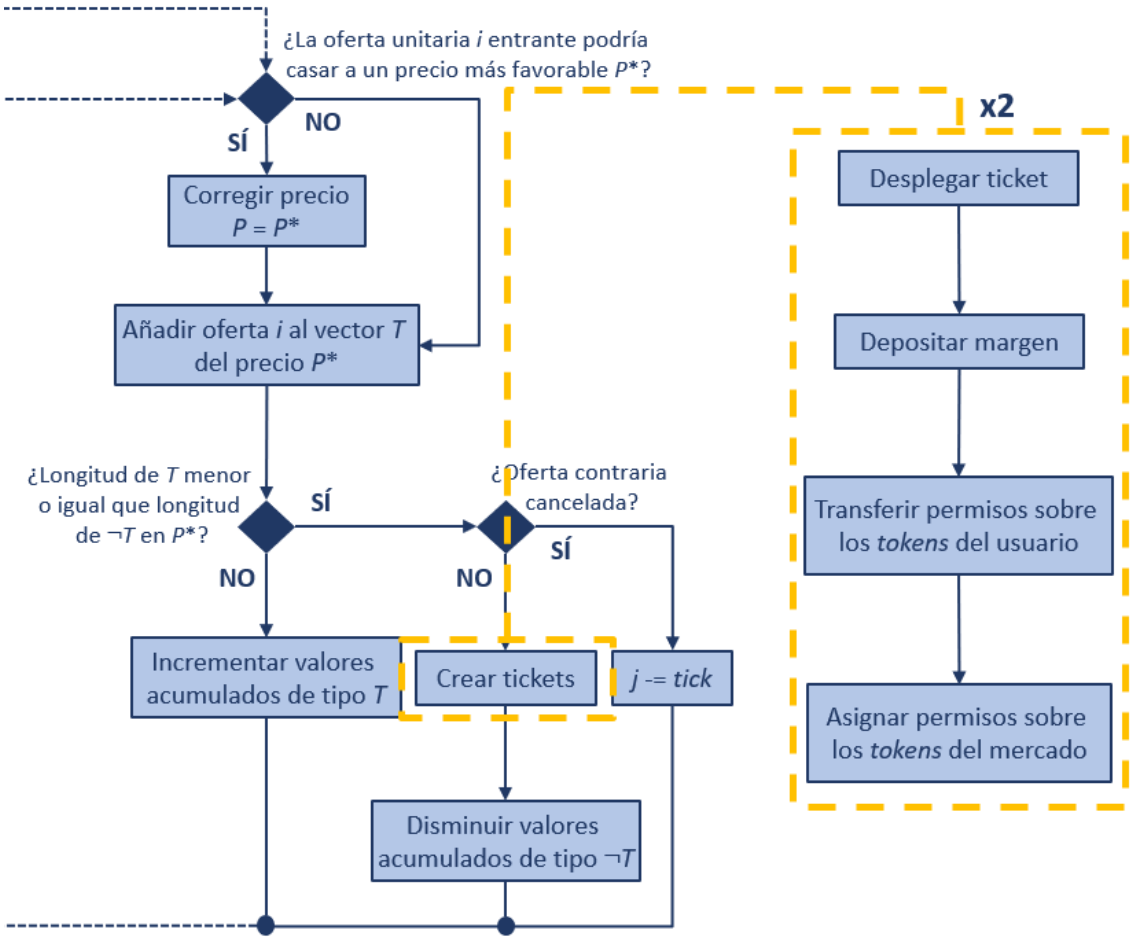


Figura 5.18. Secuencia detallada de operaciones en la creación de tickets.

En primer lugar, se aprovecha el permiso de gasto asignado al contrato mercado por parte del agente beneficiario para depositar una pequeña cantidad, que denominaremos margen, en el propio contrato ticket. Después, se transfiere parte del permiso de gasto al ticket, a fin de que éste pueda retirar los fondos del agente correspondiente en caso necesario. En tercer lugar, es el contrato mercado quien debe asignar un cierto permiso al contrato ticket.

Con todo ello conseguimos que:

1) el ticket bloquee una pequeña cantidad de *tokens* del usuario como garantía y
2) que el ticket pueda mover *tokens* desde dos cuentas:

a) la de su dueño;
b) la del contrato mercado.

Tanto el margen inicial como los permisos asignados son directamente proporcionales al precio del ticket P^* y volumen del *tick* v. Los valores concretos utilizados para el caso de estudio se recogen en la Tabla 5.4.

Concepto	Fórmula	Descripción
Margen	$\frac{1}{5}vP^*$	Garantía inicial depositada en el ticket
Permiso 1	$\frac{4}{5}vP^*$	Permiso asociado al gasto de los fondos del dueño del ticket
Permiso 2	vP^*	Permiso asociado al gasto de los fondos del mercado

Tabla 5.4. Garantía y permisos asignados al ticket.

Obsérvese que para transferir la aprobación del gasto de los fondos del usuario desde el mercado al ticket se necesita obligatoriamente la función `transferAllowance` implementada sobre el estándar ERC-20. Es en este punto donde comprende el valor de la solución aportada al compararla con las dos alternativas más claras a priori:

1) La primera opción sería ceder al usuario la responsabilidad de gestionar personalmente los permisos de sus tickets. El primer problema que nos encontramos es evidente: el usuario tendría que ejecutar un número considerablemente mayor de transacciones, puesto que no sería suficiente con asignar el permiso al mercado. Por otro lado, hay que recordar que el ticket no existe en un principio, de modo que no podría transferir sus derechos hasta que se produjese la casación de una de sus ofertas.
2) Otra opción sería evitar la asignación del permiso sobre los *tokens* del usuario al ticket, limitando su capacidad a utilizar exclusivamente los *tokens* del mercado. Nuevamente se trata de una operación problemática tanto en términos de costes (ya hemos visto que el uso de intermediarios redunda en un mayor número de transacciones) como a nivel operacional (sería más complicado acceder directamente a los *tokens* del usuario para gestionar las garantías).

A partir de la generación del ticket, éste se verá sometido a un proceso diario de actualización de estado hasta que se finalice el período de entrega del futuro, momento en el cual se hará efectiva la liquidación y se destruirá el contrato.

5.4.2 *Mark to Market*

La ejecución del contrato de futuros consiste en una serie de liquidaciones diarias de pérdidas y ganancias (*Mark to Market*) divididas en dos períodos definidos en el propio mercado:

- Negociación
- Entrega

Para exponer el funcionamiento de este sistema, lo más sencillo es utilizar un pequeño ejemplo.

Vamos a suponer que A (comprador) y B (vendedor) negocian la compraventa de un futuro de electricidad consistente en la entrega de un volumen diario de 1 MWh a un precio de 50 €/MWh durante los días 4, 5 y 6 lo cual supone un precio total de 150 €. La negociación se produce en el día 0, de modo que los días 1, 2 y 3 constituyen el período de negociación. Con el paso de los días el precio de la electricidad fluctúa, dando lugar a la serie de precios de la Figura 5.19. Por simplicidad consideraremos el precio medio diario.

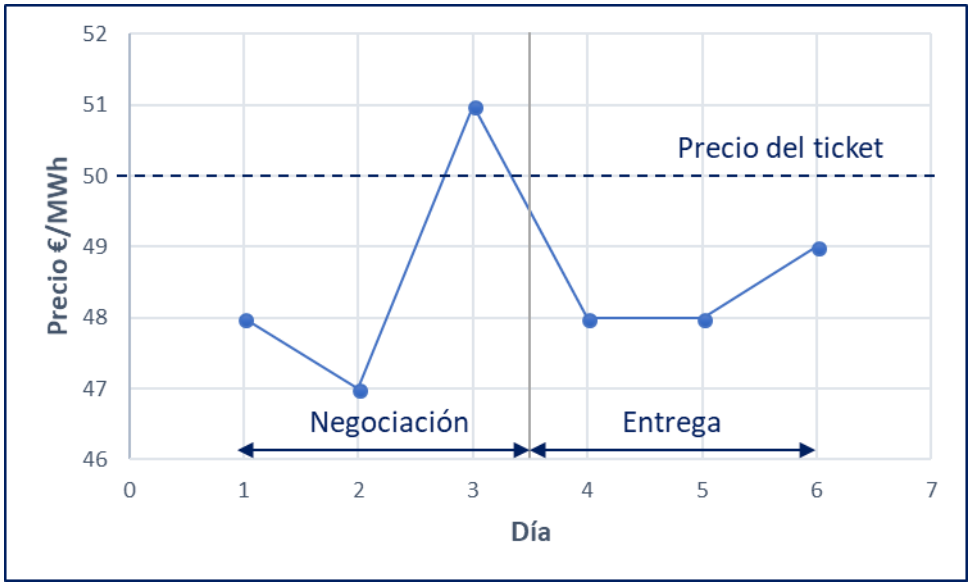


Figura 5.19. Ejemplo de liquidación diaria de PyG: secuencia de precios.

Si observamos la secuencia, la caída del precio en los dos primeros días plantea un dilema para el comprador: ha aceptado la compra por 50 €/MWh de un bien que, de seguir la misma tendencia, tendrá un valor inferior. De hecho, a la vista de la serie completa, para A sería más favorable renunciar al contrato de futuros y acudir al mercado diario. Para evitar esta

posibilidad y garantizar el cumplimiento del acuerdo se utilizan las liquidaciones diarias de pérdidas y ganancias.

El objetivo es que la transacción del futuro se realice al mismo precio del mercado diario, de modo que los días 4 y 5 B suministrará un volumen de 1 MWh diario a un precio de 48 €/MWh, y el día 6 hará lo propio a un precio de 49 €/MWh. Sin embargo, para que el contrato de futuros sea efectivo es preciso que diariamente se efectúe la liquidación de manera conveniente. Como se puede observar en la Figura 5.20, las transferencias diarias provocan una variación final en el balance de A de valor -5, mientras que en el caso de B esta variación resulta de +5. Por tanto, en valor neto se cumple el pago de 150 € por parte de A, cumpliendo con lo establecido en el contrato de futuros.

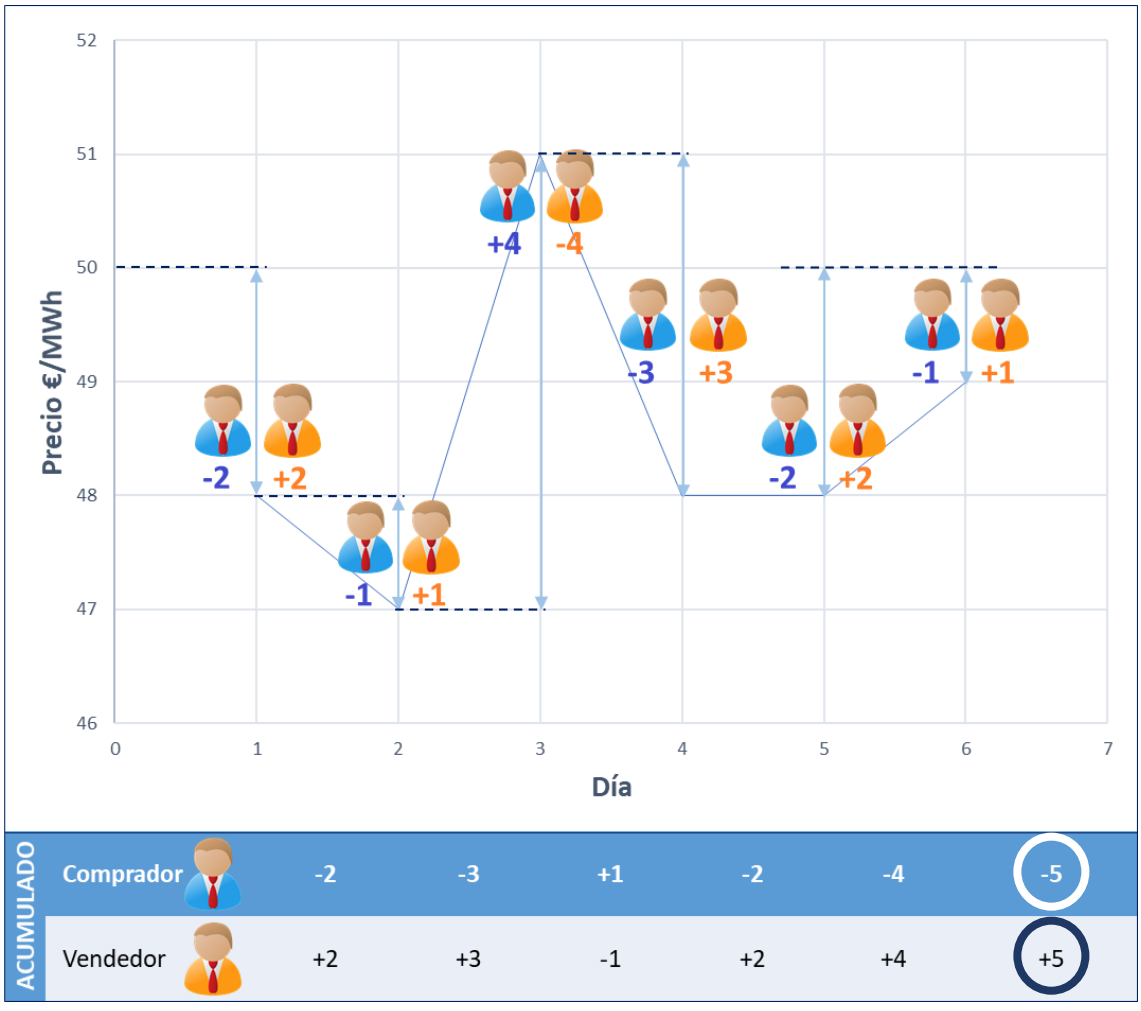


Figura 5.20. Ejemplo de liquidación diaria de PyG: actualización de balances.

En dicha figura se representan con línea discontinua las referencias tomadas para el cálculo de las liquidaciones. La referencia se calcula en cada caso como:

1) la señal del precio del día anterior, si el día actual pertenece al intervalo comprendido entre el segundo día del período de negociación y el primer día de entrega, ambos inclusive (referido como período de negociación corregido en adelante);
2) el precio del ticket (precio de casación de la oferta), en caso contrario.

La implementación del procedimiento descrito se realiza a mediante el diagrama de flujo de la Figura 5.21.

Para enviar la señal de precio al contrato ticket se utiliza el concepto de oráculo. En Blockchain, un oráculo no es más que una dirección verificada que permite volcar información del exterior en la red. En nuestro caso, por tanto, se considera que existe un oráculo que transmite de manera veraz y con periodicidad diaria el precio de la electricidad. De cara a llevar a cabo una aplicación real, lo más lógico sería que el rol de oráculo de precios fuese desempeñado por un nodo perteneciente al operador del mercado.

El algoritmo asume por defecto que el ticket es de compra (lo cual implica `buy = 1` y `sell = 0`), de modo que ha de corregir las variables utilizadas para calcular el balance del ticket en caso contrario. Es muy importante aclarar que dicho balance no representa la cantidad de *tokens* bloqueados como garantía en el ticket, sino que se trata de una variable utilizada para simular la liquidación de pérdidas y ganancias. Esta variable se inicializa al valor del margen depositado, dividido entre el volumen del *tick*. De este modo, la fluctuación del balance se corresponde directamente con la del precio de la electricidad, medido en *tokens* por unidad de energía. Para obtener las cantidades reales a liquidar, simplemente hay que escalar el balance al volumen del *tick*.

Se emplea una variable en lugar de efectuar las transferencias correspondientes puesto que el movimiento de *tokens* en cada actualización supondría un coste mucho más elevado en términos de gas. Además, el número de transferencias requerido conduciría en muchos casos al consumo total del permiso del ticket, lo cual supondría un problema.

Sí se ejecuta la transferencia de *tokens*, en cambio, cuando el balance del ticket desciende por debajo de cero. En este caso, se considera necesario ampliar la garantía, para lo cual se resetea el balance y se transfiere la cantidad correspondiente al ticket.

El balance del ticket se actualiza de la manera siguiente:

```
ticketBalance += int(sell * (priceReference - _energyPrice)
+ buy * (_energyPrice - priceReference));
```

Con esta técnica conseguimos modificar el balance con la misma sentencia, independientemente de si el ticket es de compra o venta. Lo que se hace en cada actualización es incrementar o decrementar la diferencia entre la señal de precio enviada por el oráculo y la referencia establecida.

Como se puede extraer de la fórmula anterior, el incremento en el balance del ticket será positivo para el comprador siempre que la señal de precio se encuentre por encima de la referencia, ocurriendo a la inversa para el vendedor. Lógicamente, dichos incrementos serán siempre iguales en valor absoluto para comprador y vendedor.

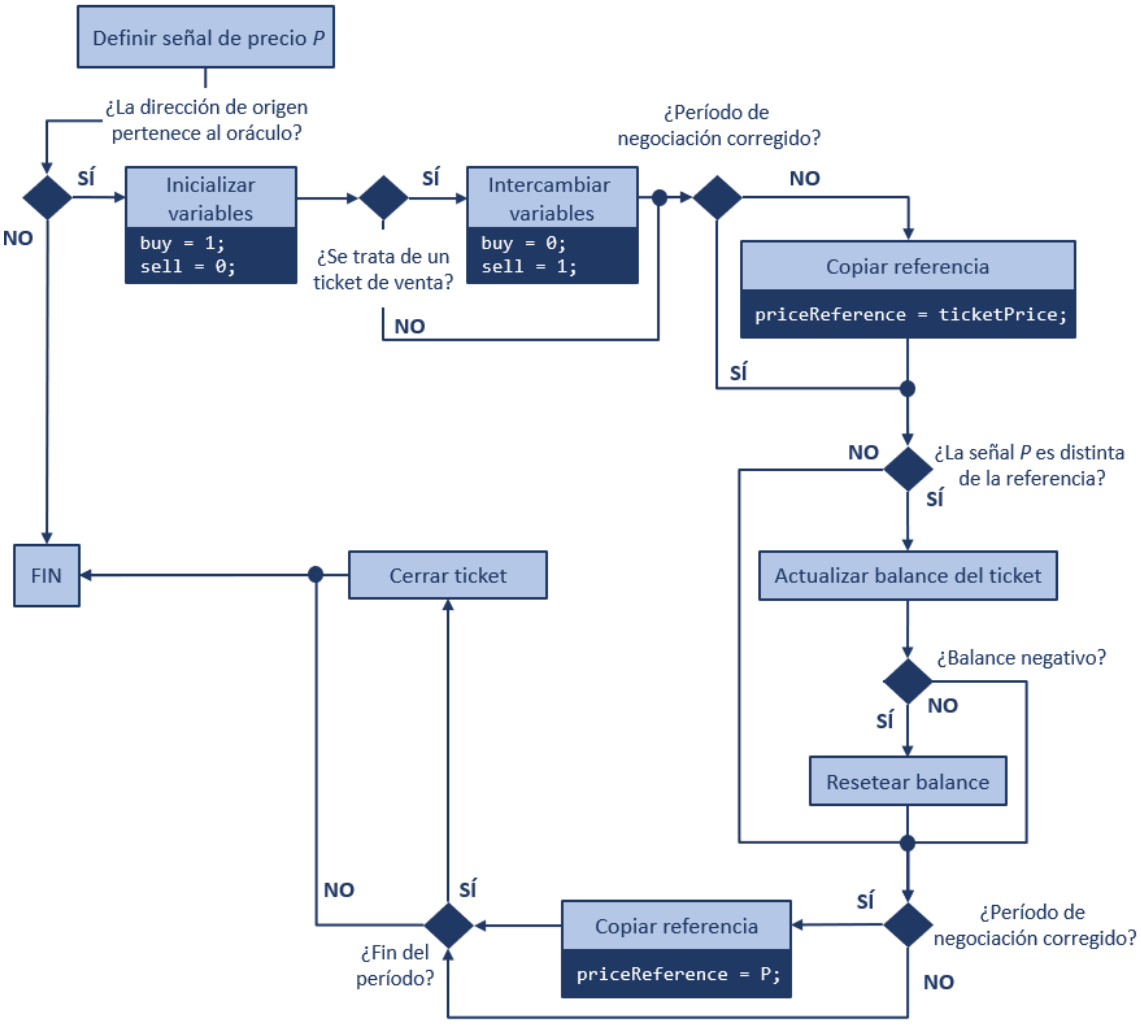
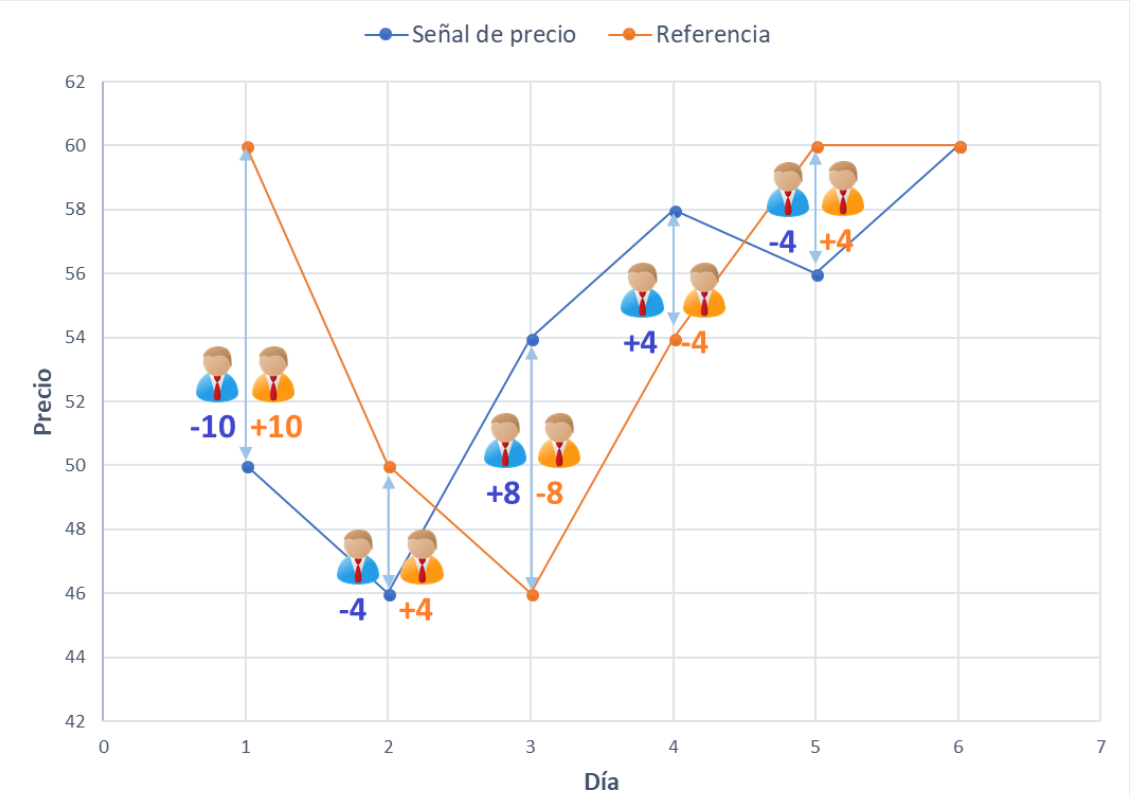


Figura 5.21. Actualización del balance del ticket.

Para mayor claridad se propone el ejemplo de la Figura 5.22, donde se ha considerado una casación a un precio de 60 *tokens*/ud. de energía en un mercado de volumen del *tick* unitario. Se supondrá un período de negociación de tres días (1, 2 y 3), al igual que en el ejemplo de la Figura 5.20. En la parte superior se ha representado la secuencia de precios, así como los valores tomados como referencia para la actualización del balance (en la tabla inferior se han sombreado los días correspondientes al período de negociación corregido).



Período		Negociación					Entrega		
Día		0	1	2	2*	3	4	5	6
Señal de precio		-	50	46	46	54	58	56	60
Referencia		-	60	50	50	46	54	60	60
Comprador	*Cuenta*	-12	-12	-12	-26	-26	-26	-26	-26
	Garantía	12	12	12	26	26	26	26	26
	Balance	12	2	-2	12	20	24	20	20
	Neto	**0**	**-10**	**-14**	**-14**	**-6**	**-2**	**-6**	**-6**
Vendedor	*Cuenta*	-12	-12	-12	-12	-12	-12	-12	-12
	Garantía	12	12	12	12	12	12	12	12
	Balance	12	22	26	26	18	14	18	18
	Neto	**0**	**10**	**14**	**14**	**6**	**2**	**6**	**6**

Figura 5.22. Ejemplo de liquidación del contrato de futuros.

Es importante observar que la garantía (*tokens* depositados en el ticket) permanece constante, y sólo se incrementa cuando se detecta un valor

negativo en el balance del ticket (día 2 en el ejemplo). En ese caso, el ticket extrae los *tokens* necesarios de la cuenta de su dueño para resetear el balance a su valor inicial (día 2*). Puede comprobarse que el valor neto permanece constante en esta operación.

El último día del contrato tiene lugar la liquidación efectiva del mismo. Para ello, ambos tickets devuelven los fondos almacenados al contrato mercado y desde éste se redistribuyen a los usuarios conforme al balance final del ticket. El flujo de *tokens* resultante en este ejemplo sería el siguiente:

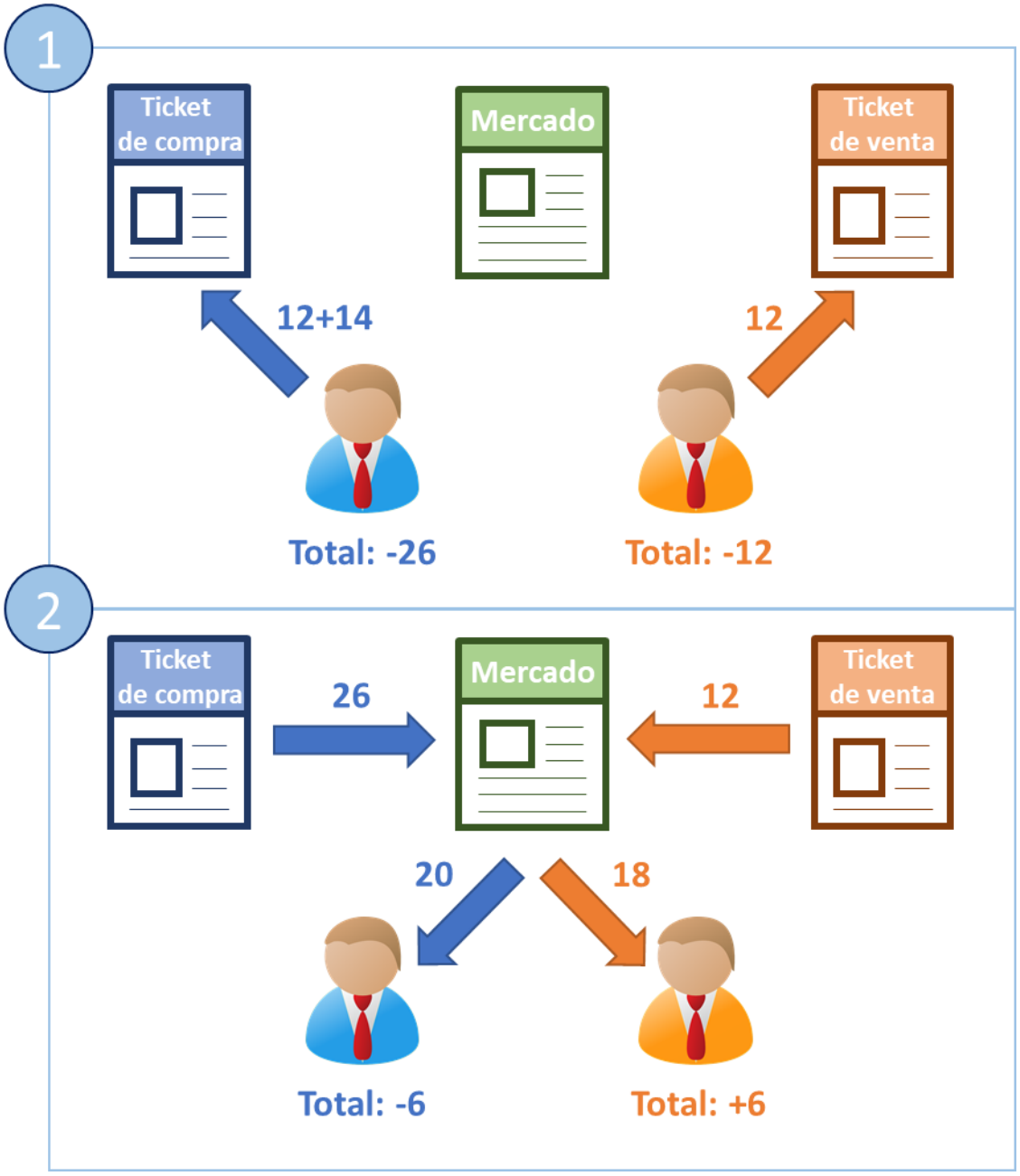


Figura 5.23. Ejemplo de flujo de valor en el mercado. Los tickets utilizan el permiso adquirido para extraer las garantías de las cuentas correspondientes (1). Los tickets devuelven las cantidades resultantes de la liquidación a través del contrato mercado al finalizar el contrato de futuros (2).

A la vista del ejemplo se puede deducir fácilmente por qué es necesario utilizar el contrato mercado como intermediario: el ticket de venta no dispone de los *tokens* suficientes para liquidar el contrato.

Liquidadas las cantidades correspondientes los tickets son destruidos, puesto que su función ha terminado.

En términos netos lo que se ha producido, en realidad, es un intercambio entre comprador y vendedor correspondiente al sumatorio de las diferencias entre el precio real de la electricidad y el precio registrado en el contrato de futuros en los días estipulados, de tal manera que se produce un resultado financieramente equivalente a la compraventa del activo en las condiciones deseadas por las partes acordantes.

5.4.3 Venta

El prototipo de mercado de derivados contempla la posibilidad de la venta de tickets, esto es, la transferencia del ticket a otro usuario antes de que el período de entrega haya comenzado. El objetivo de este desarrollo es minimizar en lo posible la generación de contratos de este tipo: si un usuario quiere, en un momento dado, cambiar su estrategia (de compra a venta o viceversa), no necesita generar un nuevo ticket de tipo contrario, sino que será suficiente con vender el que tiene en propiedad. Para ello, simplemente deberá ejecutar la función específica diseñada para tal efecto del contrato ticket en cuestión, y el sistema se encargará de colocar la oferta correspondiente en el mercado.

La mejor forma de ver cómo se gestiona esta situación es mostrar un ejemplo. Para ello, vamos a tomar nuevamente el caso del apartado anterior. Consideraremos las mismas señales de precio e idéntico precio de negociación. Sin embargo, vamos a suponer que esta vez el comprador (A), a la vista de la caída de precios, decide inteligentemente vender su ticket. Supongamos también que la casación de la oferta correspondiente se produce finalmente a un precio de 62 *tokens*/ud. de energía, transfiriendo el ticket a un segundo comprador (B).

En el momento previo al cambio de propiedad, el ticket en cuestión posee en su cuenta una garantía de 26 *tokens* pertenecientes al usuario A, y un balance de 20 (ver Figura 5.24). Una opción sería devolver desde el propio ticket los *tokens* a la dirección de A, y modificar el balance del ticket conforme a la nueva situación; sin embargo, sería preciso además transferir 6 *tokens* desde la cuenta de B a la cuenta de A debido a la diferencia de

precios entre primera y segunda negociación. Como esta segunda transferencia es (en general) necesaria, es más eficiente aprovechar esta operación para ajustar los balances de las cuentas de A y B[13].

Utilizar un solo movimiento de *tokens* de B a A tiene un inconveniente: los 26 *tokens* bloqueados como garantía deben permanecer en el ticket, y consecuentemente el balance del ticket debe permanecer invariable. Por tanto, para que el intercambio sea justo la transferencia debe tener un valor total de 32 *tokens* (26 correspondientes a la garantía que el ticket posee más los 6 *tokens* correspondientes a la diferencia de precio con respecto a la negociación original).

En resumen, el nuevo usuario paga siempre por la garantía depositada en el ticket que adquiere, más (o menos) el producto del número de días del período de entrega por la diferencia de precios entre la casación original y la nueva. En caso de que el nuevo precio sea igual al precio original registrado en el ticket, la transferencia será directamente igual a la cantidad de *tokens* almacenados en dicho ticket.

Período		Negociación					Entrega		
Día		0	1	2	3	3*	4	5	6
Señal de precio		-	50	46	54	54	58	56	60
Referencia		-	60	50	46	46	54	60	60
Comprador A	*Cuenta*	-12	-12	-26	-26	6	6	6	6
	Garantía	12	12	26	26	-	-	-	-
	Balance	12	2	12	20	-	-	-	-
	Neto	**0**	**-10**	**-14**	**-6**	**6**	**6**	**6**	**6**
Comprador B	*Cuenta*	-	-	-	-	-32	-32	-32	-32
	Garantía	-	-	-	-	26	26	26	26
	Balance	-	-	-	-	20	24	20	20
	Neto	**-**	**-**	**-**	**-**	**-12**	**-8**	**-12**	**-12**
Vendedor	*Cuenta*	-12	-12	-12	-12	-12	-12	-12	-12
	Garantía	12	12	12	12	12	12	12	12
	Balance	12	22	26	18	18	14	18	18
	Neto	**0**	**10**	**14**	**6**	**6**	**2**	**6**	**6**

Figura 5.24. Ejemplo de liquidación del contrato de futuros con venta de ticket.

[13] En este caso, el usuario A acuerda la compra de 1 unidad de energía/día (recordemos que se asume volumen del *tick* unitario en el ejemplo) a un precio de 60 *tokens*/día durante un período de tres días, lo que hace un precio total de 120 *tokens*. El usuario B hace lo propio por un precio total de 126 *tokens*, de modo que A genera un beneficio neto de 6 *tokens* con la operación. Esta operación de transferencia de valor será necesaria siempre que exista una diferencia entre el precio original y el de la segunda negociación. Nótese además que el sentido del flujo dependerá del tipo de ticket (compra o venta) y de cuál de los dos precios sea superior; por ejemplo, en el caso anterior, si el segundo precio hubiese resultado inferior a 60 *tokens*, el sentido del flujo habría resultado opuesto, redundando en pérdidas para el comprador A.

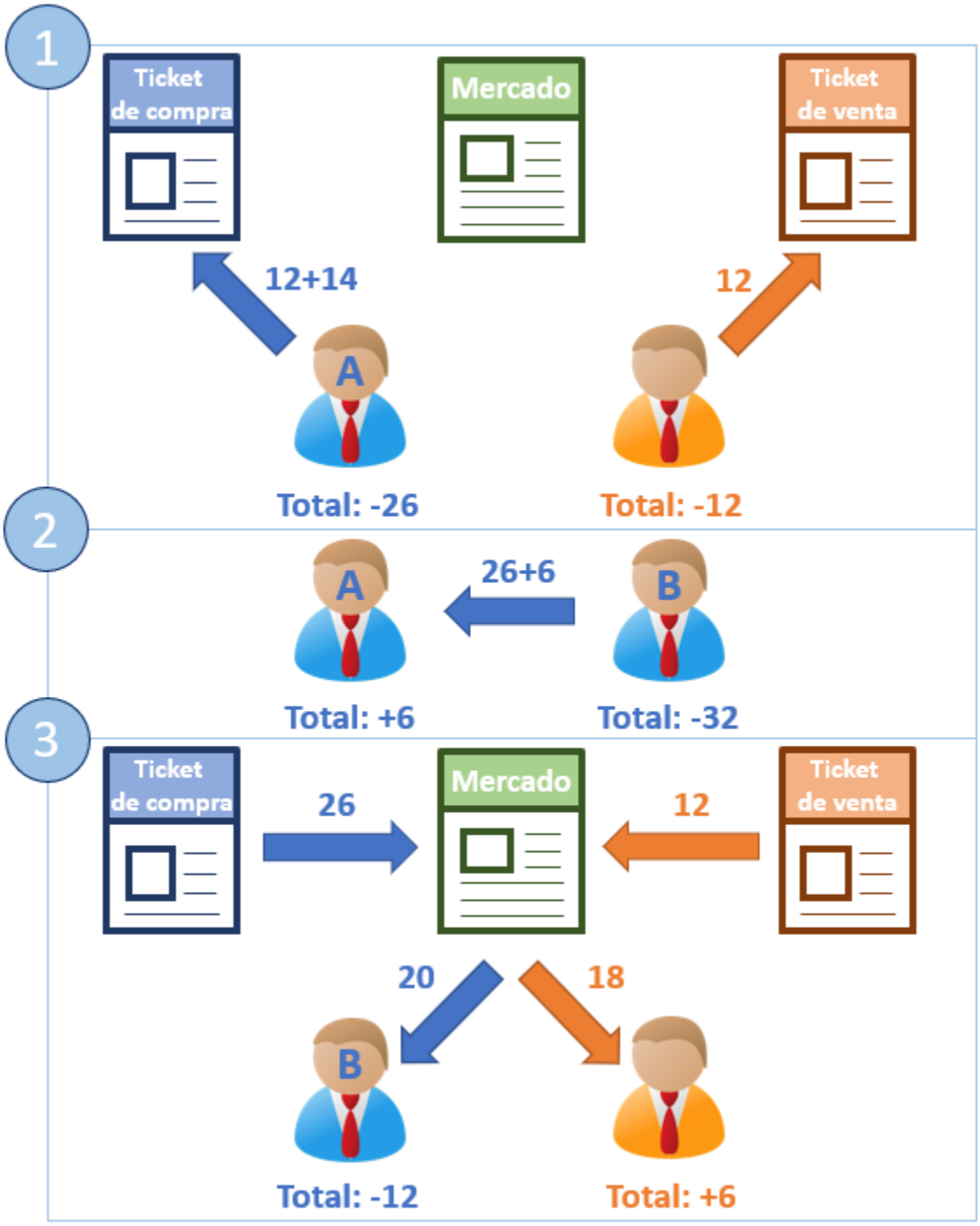


Figura 5.25. Ejemplo de flujo de valor en el mercado (caso de venta de ticket). Los tickets utilizan el permiso adquirido para extraer las garantías de las cuentas correspondientes (1). Se produce la transferencia del ticket de compra al usuario B, el cual debe pagar las cantidades requeridas al comprador original (2). Los tickets devuelven las cantidades resultantes de la liquidación a través del contrato mercado al finalizar el contrato de futuros (3).

A la vista de los resultados se puede comprobar que los tres agentes obtienen las cantidades que les corresponden, satisfaciendo así los términos del contrato de futuros. Es interesante también reseñar que, lógicamente, el sumatorio de ganancias de los tres agentes involucrados será siempre nulo,

puesto que no se crean ni destruyen *tokens* en el proceso; lo que ganan unos es lo que pierden otros.

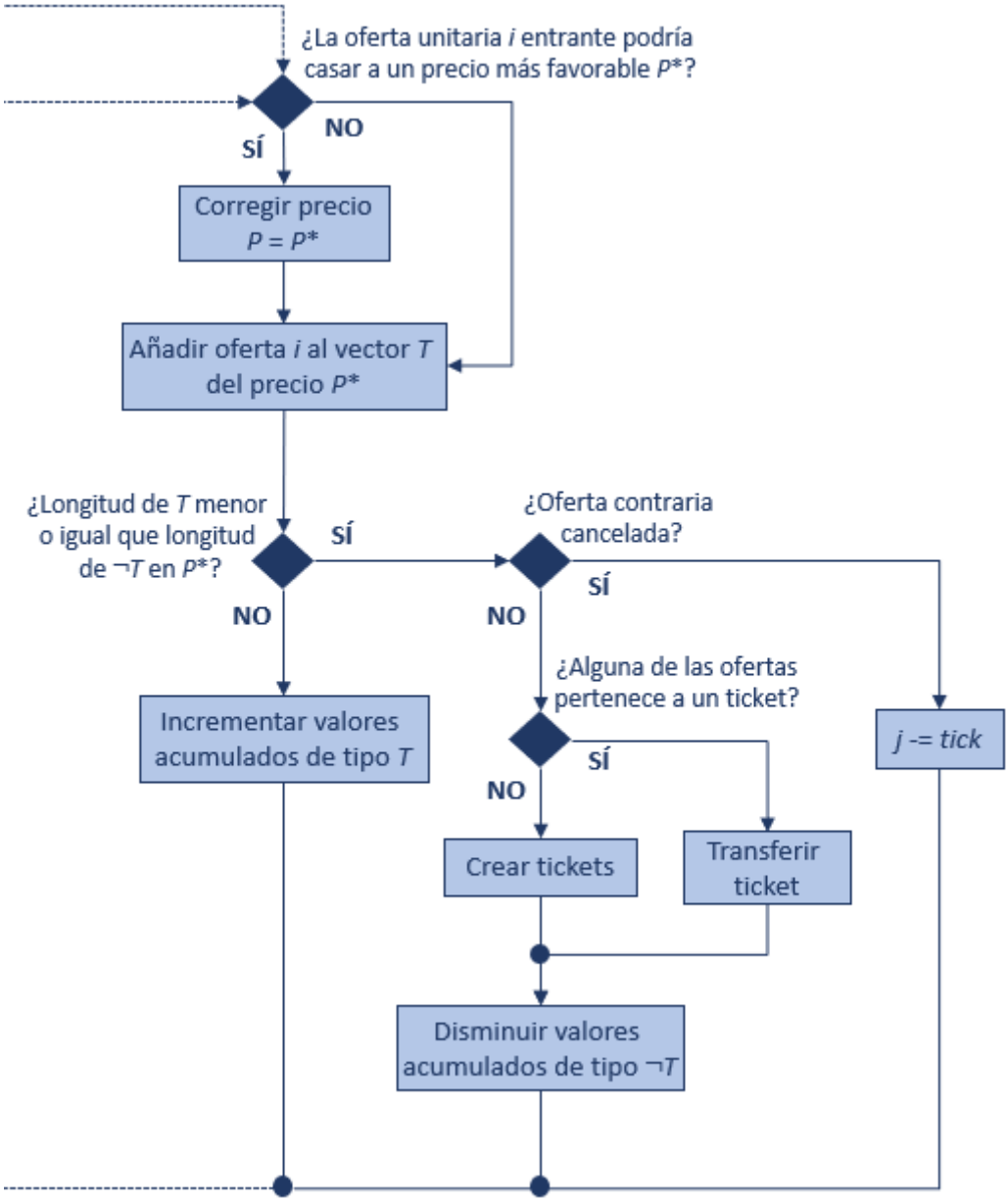


Figura 5.26. Ampliación del flujograma de casación.

Una vez explicada la metodología adoptada en caso de que se produzca la venta de un ticket, cabe preguntarse cómo se identifica dicha situación. La respuesta reside una vez más en el algoritmo de casación. Se utiliza una idea similar a la de almacenamiento de ofertas casadas (ver 5.3.3), sólo que en este caso es incluso más sencillo puesto que no se dispone de una matriz sino de una lista:

```
mapping (address => bool) public isTicket;
```

Cuando se genera un ticket, la variable `isTicket[T]` se pone automáticamente a 1, siendo *T* la dirección del nuevo contrato desplegado. De esta forma, `isTicket[msg.sender]` devolverá `true` si la dirección de origen de la transacción es un ticket, y `false` en caso contrario. Esta comprobación constituye una ampliación del algoritmo de casación (ver Figura 5.26), no incluida hasta el momento por simplicidad.

Finalmente, cabe indicar que la funcionalidad añadida de venta del ticket se consigue en parte gracias a los contratos **Owned** y **OneUserOneOwner** (Figura 5.27), los cuales se han pasado por alto hasta este instante. Sin entrar en profundidades, estos contratos se utilizan para definir de manera sencilla dos elementos fundamentales del sistema de venta: la dirección del usuario y la dirección del dueño del contrato.

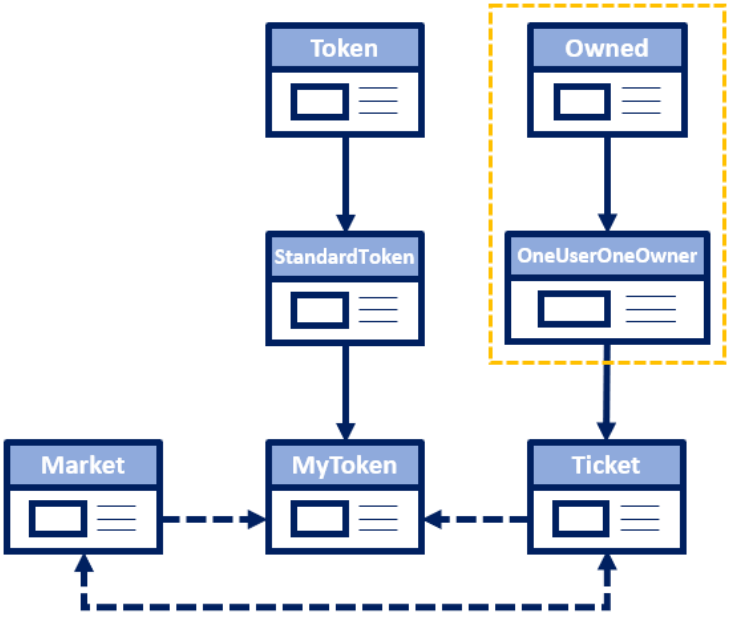


Figura 5.27. Estructura de contratos del prototipo de mercado de derivados (énfasis en los contratos "padre" del contrato ticket).

La distinción previa es muy importante, ya que en la programación del modelo se entiende por dueño del contrato ticket el propio contrato mercado, mientras que la dirección del comprador/vendedor se almacena como usuario del contrato. Nótese que el usuario no podría ser nunca dueño del contrato, puesto que la potestad de transferir el ticket (esto es, modificar la dirección del usuario) ha de pertenecer única y exclusivamente al mercado. No obstante, es imprescindible considerar la figura del usuario, puesto que es la única dirección que debe tener acceso a cierta funcionalidad del ticket (consultar el balance, ejecutar la orden de venta, etc.).

5.5 Casos no soportados por el prototipo

Para concluir el capítulo, es necesario comentar dos casos de extraña ocurrencia.

El primer caso que merece ser citado es el correspondiente a la casación de dos ofertas enviadas por un ticket (es decir, correspondientes a la ejecución de la orden de venta del ticket). Este caso no se encuentra soportado por el prototipo de mercado de derivados; sin embargo, sí se ejecuta la comprobación pertinente para, en caso de darse esta situación, revertir el estado del contrato e impedir que se ejecute la transacción de manera efectiva.

El segundo caso singular es que se produzca la casación de dos ofertas procedentes del mismo agente. Esta situación se entiende como una cancelación manual de la primera oferta.

Capítulo 6

Pruebas y resultados

El objetivo de este capítulo es presentar los resultados obtenidos a través de distintas simulaciones desarrolladas para evaluar el funcionamiento del prototipo de mercado de derivados.

Las simulaciones son tests programados en Javascript y ejecutables desde Truffle. En esencia, podemos considerar cada test como una secuencia de llamadas a las distintas funciones del modelo. En función del propósito de la simulación, los tests han sido clasificados en dos grupos:

- Pruebas. Tests específicos diseñados para validar la respuesta del sistema en situaciones concretas.
- Resultados. Simulaciones de uso del prototipo para cuantificar los costes del sistema propuesto.

6.1 Pruebas

A continuación, se listan las comprobaciones realizadas para verificar el comportamiento de los contratos.

Contrato	Test	Descripción	Resultado
Token	01	Despliegue del contrato + inicialización de variables	✓
	02	Usuario A envía *x* tokens a B	✓
	03	Usuario A aprueba el gasto de *x* tokens a B	✓
	04	B utiliza el permiso asignado por A	✓
Mercado	05	Despliegue del contrato + inicialización de variables	✓
	06	Generación de tickets (casación normal)	✓
	07	Generación de tickets (corrección de precio)	✓
	08	Descomposición en ofertas unitarias	✓
	09	Cancelación de oferta	✓
	10	Oferta cancelada no produce tickets	✓
	11	Casación con oferta emitida por ticket	✓
Ticket	12	Secuencia de señales de precio (venta)	✓
	13	Secuencia de señales de precio (compra)	✓
	14	Venta del ticket	✓

Tabla 6.1. Lista de pruebas.

Como se puede observar, todos los tests ejecutados han arrojado resultados favorables, lo que certifica el funcionamiento del prototipo conforme a especificaciones.

6.2 Resultados

Los resultados en términos de costes del sistema se pueden clasificar en función del agente sobre el que recaen. De un lado, los agentes negociadores deben asumir el coste asociado al envío de ofertas, que por la modalidad continua de negociación es coincidente con el coste de la casación. Del lado contrario, el coste más representativo que debe afrontar el operador del mercado es el correspondiente a la actualización de los tickets mediante el envío de señales de precio desde la dirección oráculo, es decir, el coste asociado a las operaciones *Mark to Market*.

El origen de estos costes reside en el concepto de *gas* de la red Ethereum.

6.2.1 Gas en Ethereum

El gas es una unidad de medida que cuantifica el uso de recursos necesarios para llevar a cabo una operación o conjunto de operaciones en Ethereum. Dicho con otras palabras, es una función de la cantidad y calidad de código que se debe ejecutar para llevar a cabo una transacción concreta. De este modo, el concepto de gas representa el coste computacional que el emisor de la transacción debe afrontar.

Cuando se produce una transacción, el coste final repercutido al emisor resulta del producto del consumo de gas y el precio del mismo. El precio del gas es la cantidad de *ether* que cuesta procesar una unidad de gas, y es función del estado de la red y de la velocidad con la que se desea que la transacción en cuestión sea ejecutada. Por tanto y lógicamente, procesar una operación compleja a la velocidad más elevada posible en un momento de congestión de la red proporcionará un coste notablemente superior al de una transacción más simple de baja prioridad en un momento en el que el factor de utilización de la red sea bajo.

La función del concepto de gas es eminentemente de protección de la plataforma: se evitan así ataques que pretendan consumir gran cantidad de recursos y colapsar la red.

6.2.2 Coste en gas de la casación

Para cuantificar el coste de la casación del mercado se han llevado a cabo simulaciones de diferente magnitud. En este texto se presentan dos casos:

1) Caso reducido: 50 ofertas (ver Figura 6.1).
2) Caso ampliado: 300 ofertas.

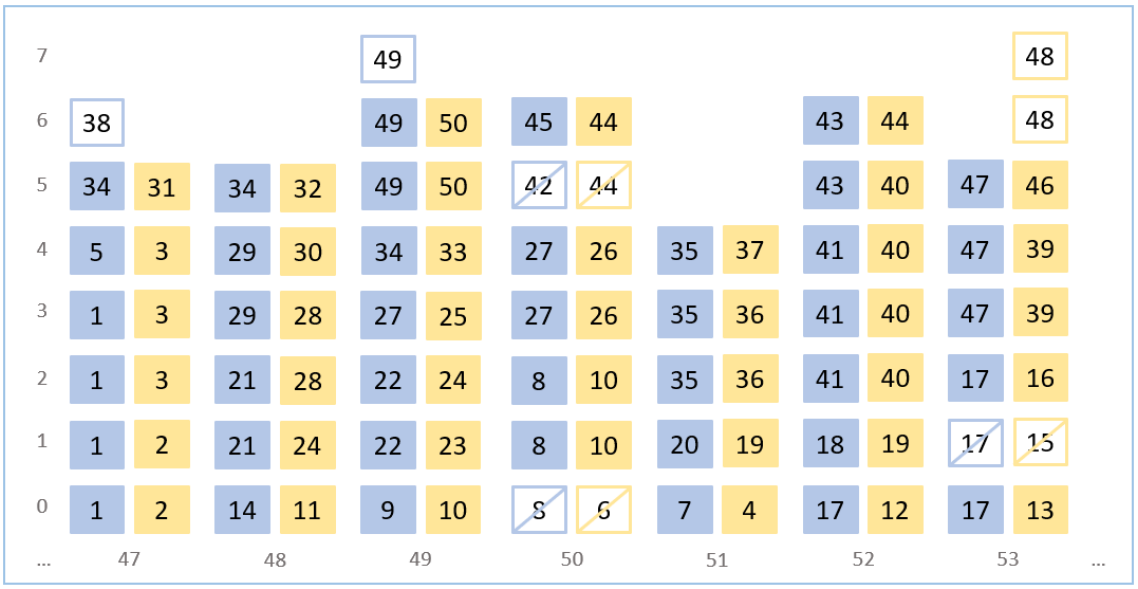


Figura 6.1. Representación del mercado (caso reducido). Los números representan el orden de envío de las ofertas.

Aunque el valor de gas utilizado en una transacción concreta es determinista, dado el elevado número de combinaciones posibles, es realmente complicado expresar el coste en función del estado del mercado (variables de las estructuras de precio, ofertas canceladas, etc.) y los parámetros de envío de la oferta. En este caso resulta más eficiente adoptar una aproximación empírica a la resolución del problema, de modo que analizaremos los datos extraídos en las simulaciones ejemplo para estimar:

1) sensibilidad del consumo de gas con respecto a las variables más significativas;
2) influencia del volumen del mercado (número de ofertas previas) en el coste de casación;
3) valores típicos de coste en gas para el envío de ofertas.

La Figura 6.2 muestra los resultados obtenidos en el caso reducido. En el eje de abscisas se ha representado el volumen del mercado anterior al envío de la oferta. El coste en gas, por su parte, se representa en el eje de ordenadas. Además, se han simbolizado las variables que previsiblemente pueden contribuir en mayor medida a explicar la variabilidad del coste, a saber:

- Cantidad total ("TotalQuantity"). Ofertas unitarias generadas con el envío, incluyendo las ofertas *dummy*.
- Creación de tickets ("TicketCreation"). Número de creaciones de tickets ejecutadas como resultado de la casación continua. La unidad

corresponde al despliegue de dos tickets (compra y venta) puesto que se generan por pares.

- Corrección de precio ("PriceChange"). Número de ofertas unitarias que han sufrido una corrección de precio.

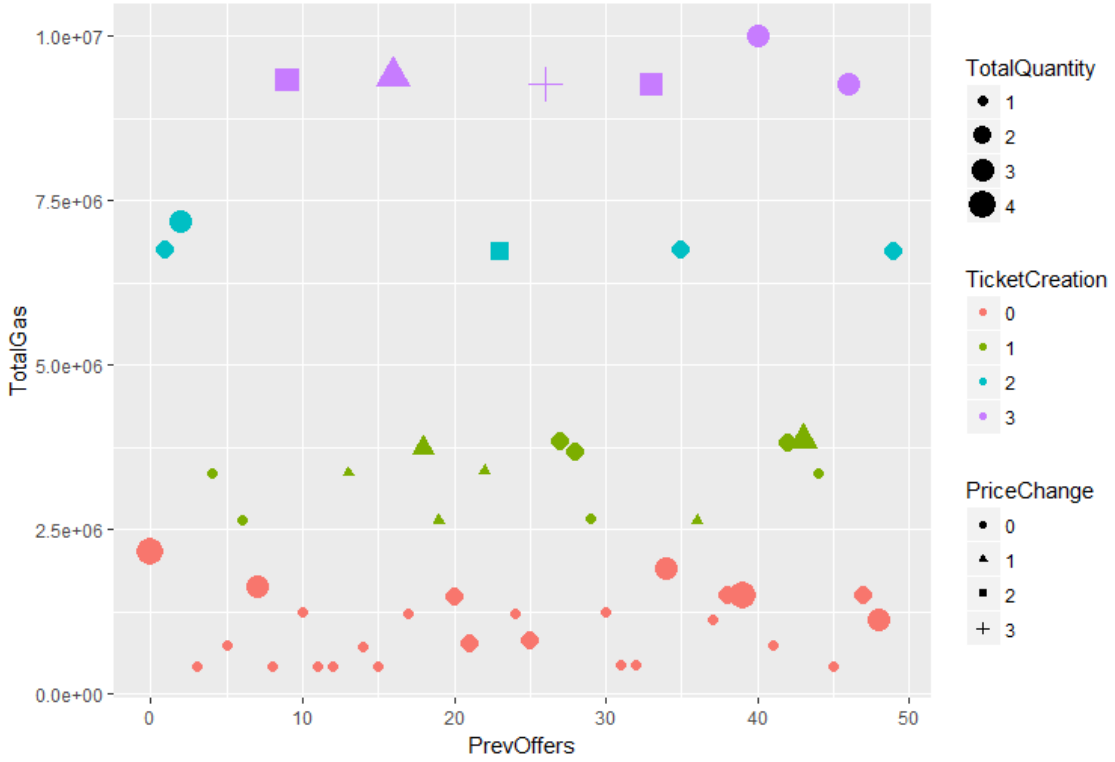


Figura 6.2. Coste total (gas) asociado al envío de una oferta en función de la cantidad de ofertas previas existentes en el mercado (caso reducido).

A la vista de los resultados obtenidos, la primera conclusión evidente es que la circunstancia más influyente en el consumo de gas es la creación de tickets. Por otro lado, no se observa un crecimiento del coste con el aumento del volumen del mercado, por lo que parece claro que el número de ofertas existentes no afecta de manera significativa al coste en gas. Ambas afirmaciones se corroboran con los resultados obtenidos en el caso ampliado (ver Figura 6.3).

Figura 6.3. Coste total (gas) asociado al envío de una oferta en función de la cantidad de ofertas previas existentes en el mercado (caso ampliado).

Se pueden distinguir claramente bandas de consumo de gas en función de las creaciones de tickets (ver Figura 6.4), de modo que es posible estimar de manera aproximada el valor de consumo de gas dado un número de tickets creados, siendo la diferencia entre los valores medios de dichas bandas de unos 3 millones. En la Tabla 6.2 se recogen los valores obtenidos en el caso ampliado; en realidad, se puede comprobar que existe un rango bastante amplio de valores dentro de cada banda de consumo de gas, lo cual es debido a otras variables que, aunque menos significativas, también influyen en el coste total.

Creación de tickets	**0**	**1**	**2**	**3**	**4**
Mínimo	276.615	2.629.874	6.523.966	9.272.563	13.039.784
Primer cuartil	591.037	3.343.142	6.611.124	9.806.239	13.142.131
Mediana	825.663	3.386.866	6.650.008	9.924.868	13.228.200
Media	874.234	3.400.230	6.746.412	9.835.409	13.242.331
Tercer cuartil	1.131.094	3.433.780	6.738.238	10.015.448	13.335.413
Máximo	2.157.879	4.455.745	7.772.224	10.722.316	13.529.275

Tabla 6.2. Valores estadísticos del consumo de gas en función del número de tickets creados.

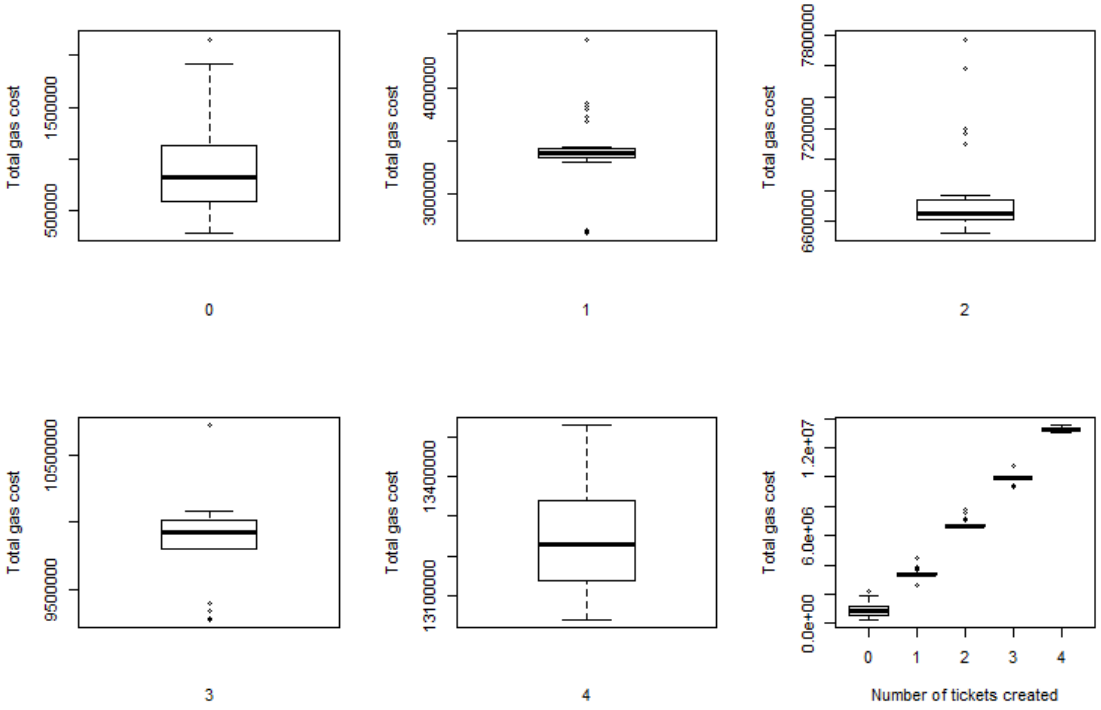


Figura 6.4. Diagramas de cajas del coste total (gas) en función del número de tickets creados a raíz del envío de la oferta.

En cuanto a los parámetros de envío de la oferta, el que afecta de manera más significativa al coste es la cantidad (ver Figura 6.5). Dentro de una misma categoría (mismo número de tickets creados), se observa un ligero incremento en el coste con la cantidad registrada, tal y como cabía esperar dado el funcionamiento del algoritmo de casación: se ejecutan tantas iteraciones del bucle como ofertas unitarias hayan sido creadas.

Ni el precio de la oferta ni el tipo parecen tener clara influencia sobre el coste. Sin embargo, sabemos que ambas variables afectan al coste total puesto que:

1) El precio de la oferta influye en el número de actualizaciones del balance acumulado. Si se añade al mercado una oferta de compra a precio P (siendo el resultado de casación negativo), se deberán incrementar los valores acumulados del precio en cuestión y todos los precios anteriores, resultando el número de operaciones $P + 1$ (incluye el precio 0). Por tanto, el coste para una oferta de compra será mayor cuanto mayor sea el precio de la misma, ocurriendo a la inversa para una oferta de venta (mayor consumo de gas a menor precio).

2) Además de lo anterior, el tipo de oferta determina si se activan o no ciertas sentencias, por lo que el coste depende necesariamente del tipo.

Lo que ocurre en el caso concreto del precio de la oferta es que los resultados son ligeramente engañosos: por el sistema de corrección de precios las ofertas más caras computacionalmente (venta a precio bajo o compra a precio alto) simplemente no se producen. Esto provoca que las ofertas más costosas se sitúen en los precios centrales (en torno al precio 50).

En cualquier caso, el elevado coste de la creación de tickets hace que el coste de estas operaciones resulte despreciable en comparación, como también ocurre con la ejecución de corrección de precio.

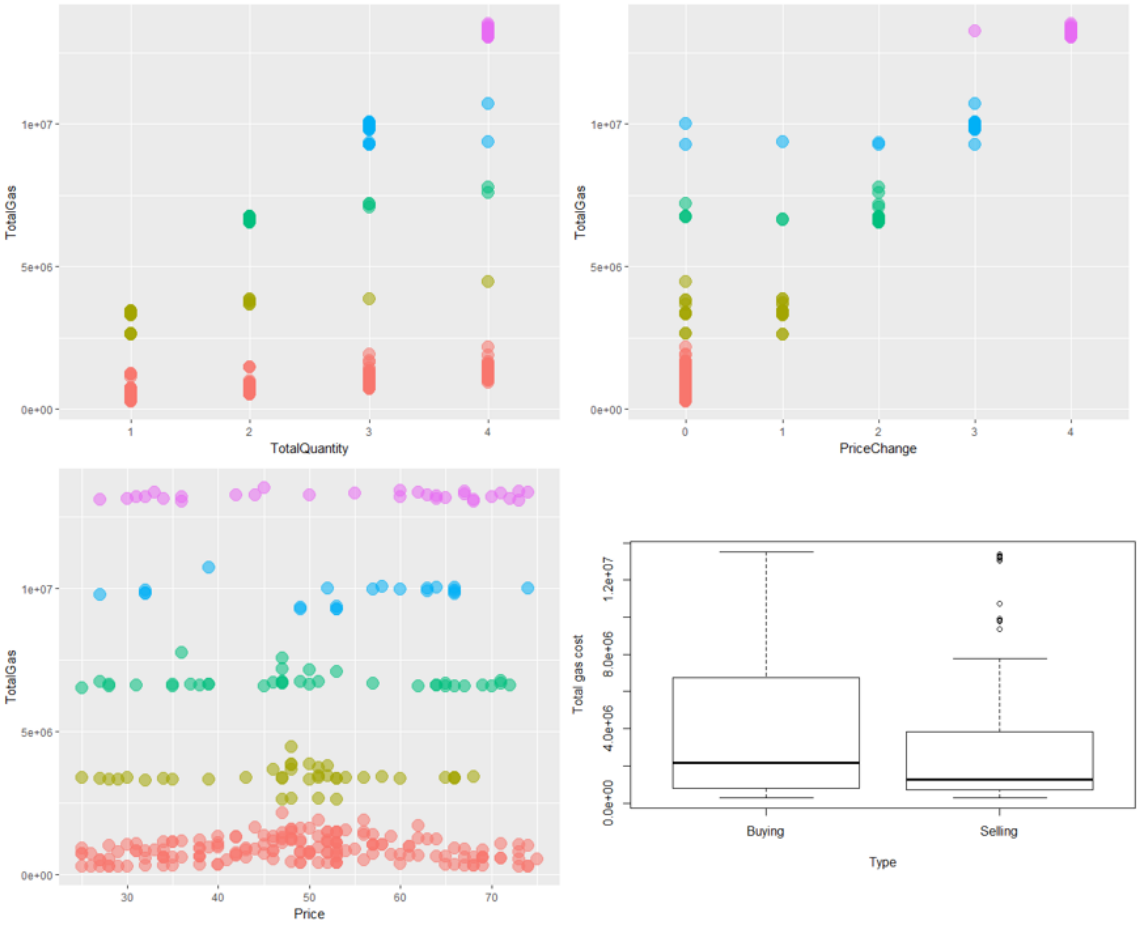


Figura 6.5. Relación entre el coste total (gas) y: (1) la cantidad de ofertas unitarias creadas (arriba-izquierda); (2) el número de correcciones de precio resultantes (arriba-derecha); (3) el precio de envío de la oferta (abajo-izquierda); (4) el tipo de oferta (abajo-derecha).

6.2.3 Coste en gas de las operaciones *Mark to Market*

El algoritmo de actualización del balance del ticket, mediante el cual se representan las liquidaciones de pérdidas y ganancias, es más simple que el algoritmo de casación, lo cual ha permitido en este caso formular el coste exacto de la operación en términos de gas.

Se han reproducido múltiples simulaciones del proceso de liquidación variando precio de casación, secuencia de señales de precio y períodos de liquidación y entrega a fin de identificar claramente de qué depende el coste en gas de actualización del balance del ticket. Con ello, se han identificado un coste base C_B y una serie de costes adicionales que se producen ante la activación de diversas situaciones:

- C_S, coste asociado al hecho de que el ticket sea de venta.
- C_{ref}, coste asociado al cambio en la referencia.
- C_{res}, coste asociado al reseteo del balance del ticket.
- C_D, coste asociado al tipo de período (se activa si el día en cuestión no pertenece al período de negociación corregido).
- C_L, coste asociado a la destrucción del ticket (último día de entrega).

Gráficamente podemos representar el coste total de la manera siguiente:

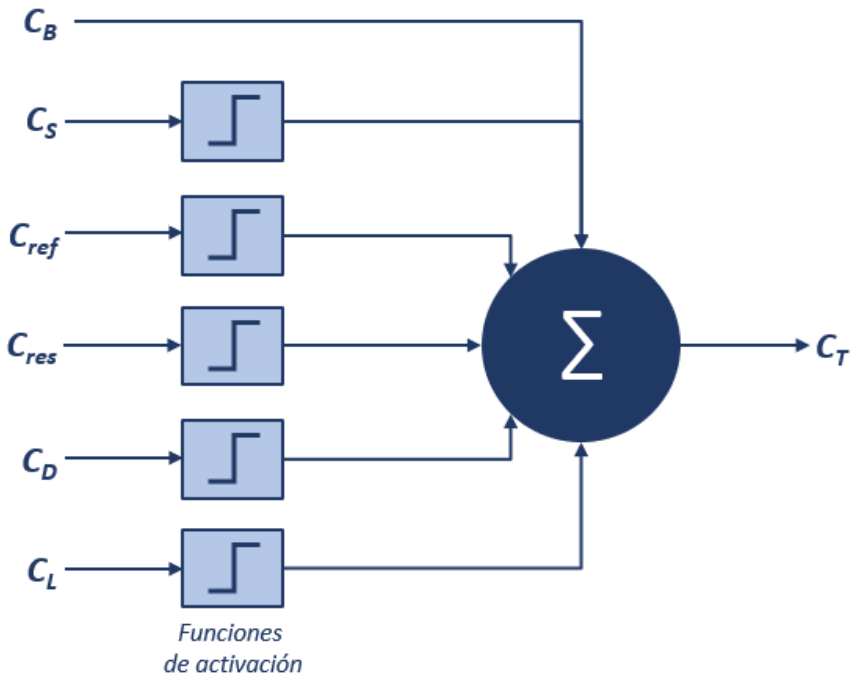


Figura 6.6. Representación de la fórmula de cálculo del coste total de actualización del balance del ticket.

Las funciones de activación actúan como un factor booleano que multiplica el coste correspondiente, valiendo 1 en caso de que se produzca el suceso y 0 en caso contrario. Los valores numéricos de los costes, por su parte, se recogen en la Tabla 6.3. Como se puede observar, los costes más importantes se producen con la secuencia de reseteo del balance del ticket, y con la destrucción del mismo.

Denominación	Valor
C_B	28430
C_S	10028
C_{ref}	6067
C_{res}	35198
C_D	200
C_L	50167

Tabla 6.3. Desglose de costes asociados a las operaciones Mark to Market.

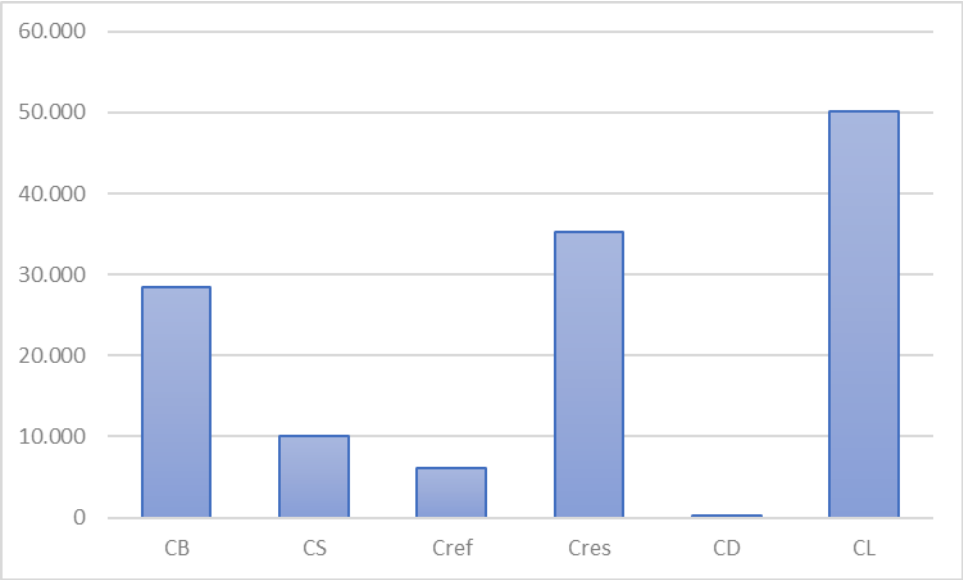
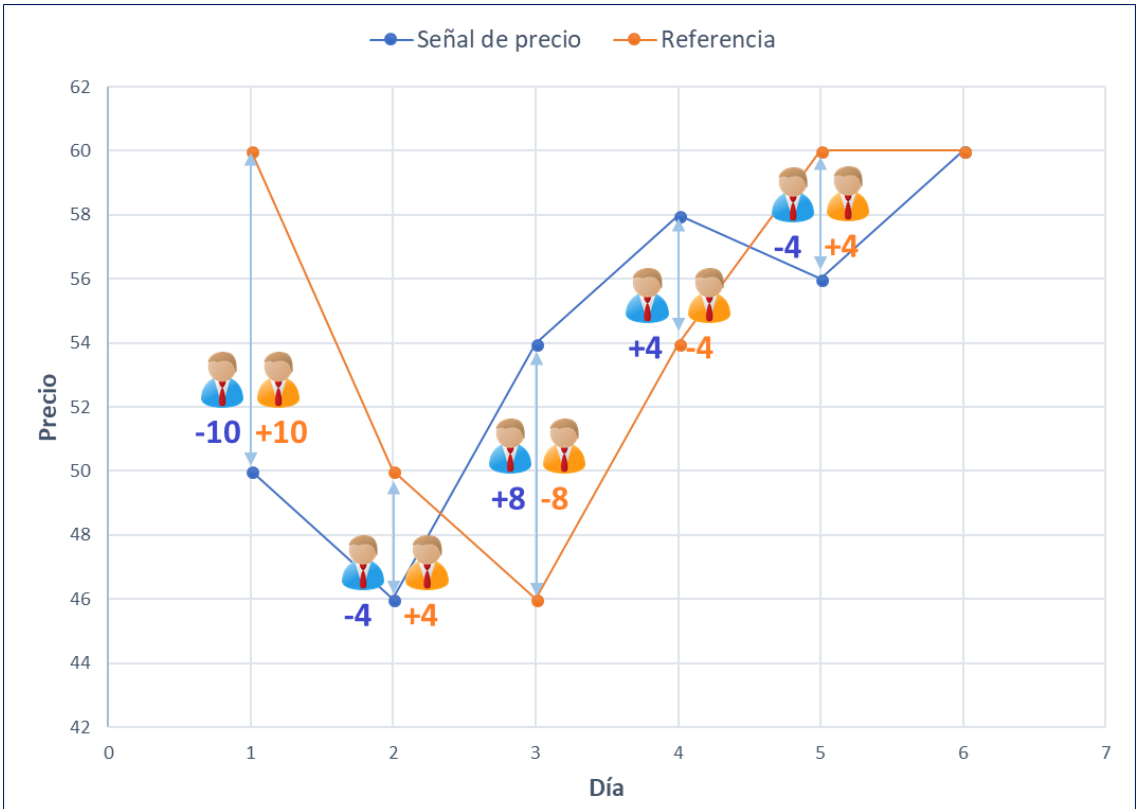


Figura 6.7. Diagrama de barras de los costes asociados a las operaciones Mark to Market.

Desde el punto de vista de la estructura del modelo, es lógico que se produzca un comportamiento como el descrito en lo que a consumo de gas respecta. Cada uno de los costes adicionales identificados se corresponden con un fragmento de código controlado por una sentencia de tipo *if*, de modo que sólo se ejecutan si se dan las circunstancias necesarias.

Nótese también que los costes no dependen de los valores numéricos concretos (por ejemplo, el precio del ticket). El coste en gas está asociado a la operación u operaciones lógicas implícitas en una sentencia y no a los operandos.

A continuación, se presenta un caso ejemplo donde se puede visualizar la evolución del coste asociado a las operaciones de liquidación. En este caso se ha creído conveniente reproducir el mismo escenario utilizado en el ejemplo del apartado 5.4.2 (ver Figura 6.8): el test ejecuta dos ofertas (compra y venta respectivamente) a precio 60, de modo que se generan los tickets correspondientes; acto seguido, se simula la secuencia de envío de señales de precio.



Período		Negociación					Entrega		
Día		0	1	2	2*	3	4	5	6
Señal de precio		-	50	46	46	54	58	56	60
Referencia		-	60	50	50	46	54	60	60
Comprador	*Cuenta*	-12	-12	-12	-26	-26	-26	-26	-26
	Garantía	12	12	12	26	26	26	26	26
	Balance	12	2	-2	12	20	24	20	20
	Neto	**0**	**-10**	**-14**	**-14**	**-6**	**-2**	**-6**	**-6**
Vendedor	*Cuenta*	-12	-12	-12	-12	-12	-12	-12	-12
	Garantía	12	12	12	12	12	12	12	12
	Balance	12	22	26	26	18	14	18	18
	Neto	**0**	**10**	**14**	**14**	**6**	**2**	**6**	**6**

Figura 6.8. Ejemplo de liquidación del contrato de futuros.

En la Figura 6.9 se muestra el coste acumulado con cada nuevo envío de señal de precio. En ella podemos observar algunos de los aspectos comentados previamente:

- El consumo de gas es más elevado si el ticket es de venta (*ceteris paribus*). En este caso, es interesante ver cómo el coste acumulado del ticket de compra supera el correspondiente al ticket de venta en el día 2. Esto es debido al elevado consumo de gas derivado del reseteo del ticket de compra, ejecutado tras la detección de un valor negativo en el balance del mismo ticket.
- En el último día de entrega el consumo de gas es notablemente superior debido a la influencia de la liquidación del contrato de futuros (recordemos que sólo se produce un movimiento efectivo de *tokens* en el instante final) y la destrucción del ticket.
- Entre los días 3 y 5, la figura parece indicar un crecimiento constante en los costes acumulados, puesto que no se aprecia ningún cambio en las pendientes de las rectas correspondientes. En realidad, durante estos días se dan períodos distintos y por tanto los costes difieren ligeramente; sin embargo, se trata de una desviación despreciable que concuerda con el bajo valor C_D. La influencia del cambio de referencia, por otro lado, no es apreciable puesto que en este intervalo no existen dos días consecutivos tales que se mantenga la referencia para el cálculo del balance.

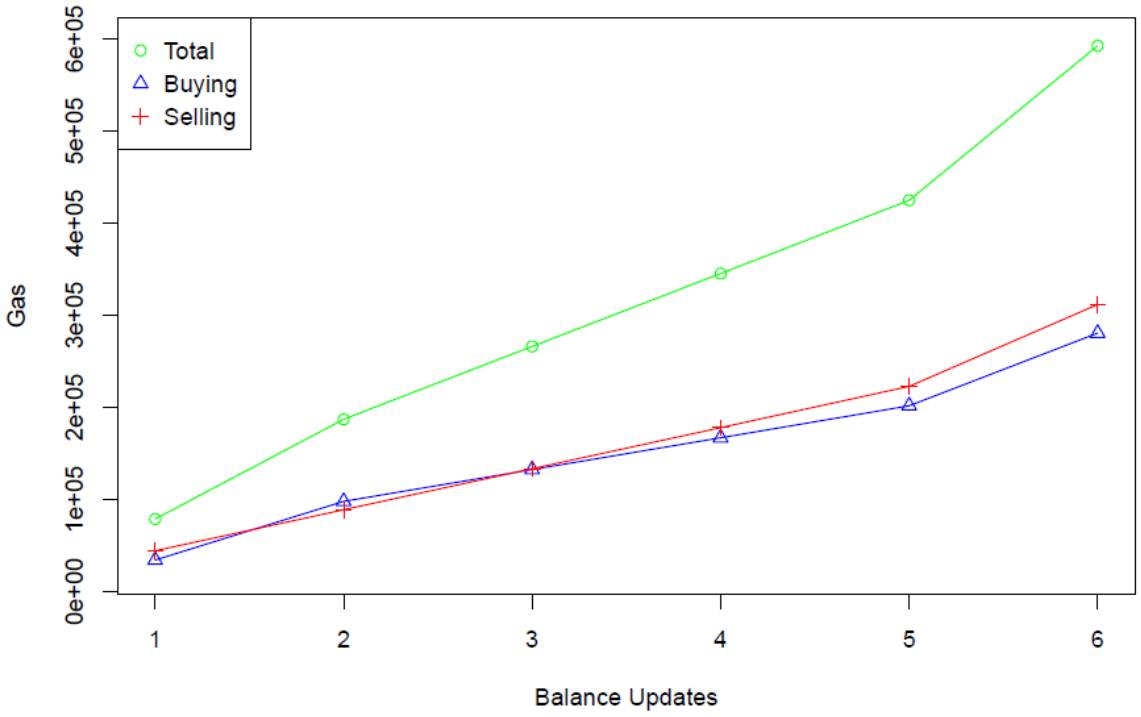


Figura 6.9. Coste acumulado (gas) asociado a la actualización de los balances de los tickets.

Como se puede comprobar, el coste en gas asociado a la operación de actualización de balance del ticket es notablemente inferior al

correspondiente al envío de ofertas, con una diferencia típica de un orden de magnitud.

Capítulo 7

Conclusiones y desarrollos futuros

Este TFM, además de proporcionar una perspectiva general de algunos de los desarrollos de Blockchain más destacados en el ámbito del mercado eléctrico, constituye una prueba de concepto que abre la puerta a la posibilidad de implementar total o parcialmente mercados financieros de casación continua en Blockchain.

En este capítulo se recogen las principales conclusiones del trabajo, y se proponen diversas líneas de investigación/desarrollo de cara a futuros estudios relacionados.

7.1 Conclusiones acerca del prototipo desarrollado

Tras la realización de las pruebas pertinentes, recogidas en el capítulo sexto, se valida el funcionamiento del prototipo de mercado de derivados en el

entorno de simulación, conforme a las especificaciones descritas en los capítulos cuarto y quinto de este texto. Los contratos que conforman el modelo han demostrado responder adecuadamente ante múltiples situaciones de diversa índole.

La validación del sistema queda supeditada, no obstante, al estudio de una casuística más extensa, así como a la respuesta del modelo a gran escala en plataforma real.

7.2 Conclusiones acerca de la aplicabilidad de Blockchain en el mercado mayorista

Los resultados obtenidos en este TFM certifican la viabilidad técnica de la integración de mercados financieros en Blockchain. La viabilidad económica, por su parte, queda en entredicho a la vista del elevado coste computacional del sistema demostrado a través de diversas simulaciones. En cualquier caso, para aportar un veredicto tajante a este respecto son necesarios estudios comparativos más exhaustivos. Sí es posible, no obstante, arrojar luz sobre algunas cuestiones específicas:

- El mayor esfuerzo computacional del sistema se encuentra en el algoritmo de casación, lo cual queda patente a través de los elevados valores de consumo de gas obtenidos en 6.2.2. Este hecho sugiere la opción de desacoplar el proceso de casación y ejecutarlo de manera externa. De cara a una posible aplicación real del sistema, ésta alternativa sería más eficiente.
- Por el mismo motivo, la opción de desarrollar aplicaciones como la propuesta en redes públicas queda totalmente descartada, en tanto que se ha demostrado ineficiente.
- De acuerdo con lo anterior, son dos las posibilidades que tienen cabida desde el prisma de la implantación real del sistema:
    1) Desarrollo de una red privada *ad-hoc*.
    2) Desarrollo de una red permisionada sobre una plataforma de propósito general como Ethereum.

En cualquiera de los dos casos anteriores existiría la posibilidad de variar el coste asociado al consumo de gas jugando con el precio del mismo, pudiendo incluso anular dicho coste. Puesto que se trataría siempre de una red permisionada/privada, no tendría demasiado sentido aplicar un coste del gas no nulo con el objetivo de asegurar la red (prevención de ataques DDoS),

pero sí podría utilizarse para establecer automáticamente un sistema de retribución al Operador del Sistema a través de la comisión obtenida por los nodos validadores (supuestos de su propiedad). En este caso sería necesario ajustar la variable de manera conveniente.

7.3 Desarrollos futuros

Finalmente, se sugieren distintas líneas de estudio que quedan abiertas a raíz de lo expuesto en este TFM. Dichos planteamientos se pueden clasificar en función del objeto de estudio, de manera similar a como se ha hecho con las conclusiones. Para ampliar la investigación se podría:

a) mejorar/ampliar la funcionalidad del modelo (dimensión técnica del problema), por ejemplo, considerando:
    i. el soporte de un mayor número de productos,
    ii. la gestión de las garantías del mercado,
    iii. la simplificación del orden de complejidad de los algoritmos utilizados,
    iv. la implementación de mercados de casación en subasta,
    v. la posibilidad de incorporar condiciones complejas en el envío de ofertas;
b) profundizar en el estudio de aplicabilidad de la tecnología (dimensión económica del problema), para lo cual sería interesante:
    i. estudiar exhaustivamente los mecanismos de operación y estructura de costes actuales del Operador del Sistema, a fin de plantear una comparativa adecuadamente,
    ii. analizar la posible implementación en otro tipo de mercados financieros,
    iii. realizar casos a gran escala en plataforma real para evaluar la velocidad de respuesta del sistema,
    iv. redefinir el rol de las entidades participantes del mercado,
    v. optimizar la arquitectura del sistema, en especial considerando la implementación parcial del modelo en servidores o plataformas externas, y el sistema de pago y retribución de los participantes,
    vi. estudiar el problema del anonimato,
    vii. analizar las repercusiones legales.

Referencias

[1] Mengelkamp, E. et al. *Designing microgrid energy markets. A case study: The Brooklyn Microgrid*. Applied Energy, 2017.

[2] Merz, M. *Potential of the Blockchain Technology in Energy Trading*. *Blockchain Technol.* Blockchain technology Introduction for business and IT managers, pp. 51–98, 2016.

[3] Grid+. *Grid+ Whitepaper*. 2017.

[4] Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008.

[5] Preukschat, A. *Criptografía asimétrica: Sistemas de Cifra con Clave Pública - Conceptos Bitcoin (II).* Disponible en:

<https://www.oroyfinanzas.com/2014/01/criptografia-asimetrica-sistemas-cifra-clave-publica-bitcoin/>

[6] *Tipos de criptografía: criptografía simétrica, criptografía asimétrica y criptografía híbrida.* Disponible en:

<http://comentariosdemislibrosfavoritos.blogspot.com.es/2016/04/tipos-de-criptografia-criptografia.html>

[7] Molero, I. *ECDSA*. Blockchain: La Revolución Industrial de Internet. 2017. Disponible en:

<http://libroblockchain.com/ecdsa/>

[8] Wüst, K; Gervais, A. *Do you need a Blockchain?* IACR Cryptology ePrint Archive, pp. 1–7, 2017.

[9] *Consenso*. Blockchain: La Revolución Industrial de Internet, 2017. Disponible en:

< http://libroblockchain.com/consenso/>

[10] Arjona, M. *Blockchain (IV). Las entrañas de bitcoin. Transacciones, criptografía y ASICs*. Disponible en:

<http://blog.elevenpaths.com/2017/04/blockchain-iv-las-entranas-de-bitcoin.html>

[11] Tuesta, D. et al. *Smart Contracts: ¿lo último en automatización de la confianza?* BBVA Research, p. 17, 2015.

[12] Cryptocurrency Market Capitalizations. Web:

<https://coinmarketcap.com/coins/>

[13] López, G. *DAO, empresas innovadoras basadas en Blockchain*. 2017. Disponible en:

<https://es.cointelegraph.com/news/dao-empresas-innovadoras-basadas-en-blockchain>

[14] Rea, A. et al. *COLONY: Technical white paper*, pp. 1–55, 2017.

[15] Sánchez, S; Álvarez, L. A. *La imprenta, Internet... ¿blockchain?* 2017. Disponible en:

<http://www.elmundo.es/economia/2017/07/16/5968b8e0268e3e491d8b45d0.html>

[16] Falcon Private Bank. *Falcon first Swiss private bank to add Ether, Litecoin and Bitcoin Cash to its current Bitcoin blockchain asset management services* [Nota de prensa]. 2017.

[17] Brown, R. G. et al. *Corda: An Introduction*. pp. 1–15, 2016.

[18] *Corda, la plataforma blockchain de los bancos y R3 será de código abierto*. Disponible en:

<https://www.fin-tech.es/2016/10/corda-la-plataforma-blockchain-codigo-abierto.html>

[19] Slock.it. Web:

<https://slock.it/index.html>

[20] Filament. *Filament Security Overview*. pp. 1–12, 2017.

[21] IBM Institute for Business Value. *Empowering the edge. Practical insights on a decentralized Internet of Things*.

[22] Arrieta, E. *El 'blockchain' encuentra su aliado en la Industria 4.0*. Disponible en:

<http://www.expansion.com/economia-digital/innovacion/2017/05/01/5901c89622601dbc348b45cb.html>

[23] Ponton. Gridchain. *Blockchain-based process integration for the smart grids of the future*. Disponbile en:

<https://enerchain.ponton.de/index.php/16-gridchain-blockchain-based-process-integration-for-the-smart-grids-of-the-future>

[24] Consensys. Web:

<https://consensys.net/ventures/joint-ventures/>

[25] Power Ledger. *Power Ledger White Paper*. p. 34, 2017.

[26] Power Ledger. *Platform-Token Interactions*. 2017.

[27] Enerchain. Web:

<https://enerchain.ponton.de/>

[28] Ponton. *European Energy Trading Firms test Peer-to-Peer Trading over the Blockchain* [Nota de prensa]. 2017. Disponible en:

<http://www.ponton.de/downloads/enerchain/Enerchain-PressRelease-02.05.2017_final.pdf>

[29] WePower. Web:

<https://wepower.network/smart-energy-trade>

[30] SolarCoin. Web:

<https://solarcoin.org/>

[31] Mihaylov, M. et al. *NRG-X-Change - A Novel Mechanism for Trading of Renewable Energy in Smart Grids*. Proceedings of the 3rd International Conference on Smart Grids and Green IT Systems, pp. 101–106, 2014.

[32] Mihaylov, M. et al. *Boosting the Renewable Energy Economy with NRGcoin*. 4th International Conference on ICT for Sustainability (ICT4S 2016), pp. 299–230, 2016.

[33] Mihaylov, M. et al. *SCANERGY: a Scalable and Modular System for Energy Trading Between Prosumers (Demonstration).* 14th International Conference on AAMAS 2015, pp. 1917–1918, 2015.

[34] Rocky Mountain Institute. *Energy companies join forces with Rocky Mountain Institute and Grid Singularity to launch global blockchain initiative for energy* [Nota de prensa]. Disponible en:

<https://www.rmi.org/about/news-and-press/press-release-energy-web-foundation-launch/>

[35] Energy Web Foundation. Web:

<https://energyweb.org/>

[36] ME SOLShare. Web:

<https://www.me-solshare.com/>

[37] CoinDesk. *How Bitcoin Brought Electricity to a South African School*. Disponible en:

<https://www.coindesk.com/south-african-primary-school-blockchain/>

[38] Besnainou, J. *From the Brooklyn Microgrid to EXERGY - A Conversation with Lawrence Orsini, CEO of LO3 Energy*. Cleantech, 2018. Disponible en:

<https://www.cleantech.com/from-the-brooklyn-microgrid-to-exergy-a-conversation-with-lawrence-orsini-ceo-of-lo3-energy/>

[39] Energía y Sociedad. *Manual de la Energía. El mercado mayorista*. Disponible en:

<http://www.energiaysociedad.es/manenergia/6-5-mecanismos-de-ajuste-de-demanda-y-produccion/>

[40] López, D. *Modelos para el análisis dinámico y la predicción a corto plazo de los precios de la electricidad en mercados liberalizados*. Tesis Doctoral. Universidad Politécnica de Madrid. p. 200, 2015.

[41] MIBEL. Web:

<http://www.mibel.com/index.php?lang=pt%5Cr>

[42] OMIE. *El mercado ibérico diario e intradiario*. pp. 1–10.

[43] OMIE. Web:

<http://www.omie.es/inicio/>

[44] OMIP. Web:

<https://www.omip.pt/es/>

[45] OMIP. *Reglamento de Negociación*. 2016.

[46] OMIClear. *Reglamento de Compensación*. pp. 1–6, 2009.

[47] Swan, M. *Blockchain. Blueprint for a New Economy*. 2015.

[48] Hyperledger Architecture Working Group. *Hyperledger Architecture, Volume 1*. Hyperledger.org, vol. 1, p. 15, 2017.