ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Máster

ANÁLISIS Y OPTIMIZACIÓN DE UN MARKETPLACE PARA LA INTERNET DE LAS COSAS BASADO EN BLOCKCHAIN

(Analysis and optimization of a Blockchain-based Marketplace for the Internet of Things)

Para acceder al Titulo de

Máster Universitario en Ingeniería de Telecomunicación

Autor: Víctor González Carril

Septiembre - 2021

Resumen

Es innegable que el Internet de las Cosas tiene cada vez más influencia sobre el mundo tecnológico. Tanto la generación de datos de carácter masivo como su análisis han probado ser recursos de gran utilidad, suscitando interés en la investigación y desarrollo de los denominados mercados de datos. En este contexto, en este Trabajo Fin de Máster (TFM) se ha analizado una plataforma basada en *Blockchain*, denominada *Blockchain-based IoT Data Darketplace* (BIDM), en la que productores IoT y consumidores de datos puedan negociar el intercambio de medidas en un entorno descentralizado, seguro y transparente. Además, en el TFM también se ha hecho una revisión del estado del arte para poner en contexto el BIDM y evaluar sus principales aportaciones, así como identificar conceptos y funcionalidades, que hayan recibido poca atención. Posteriormente, se plantea una extensión del BIDM para soportar diferentes modos de operación, específicamente un modelo de suscripciones a flujos de datos. La evaluación que se ha realizado del comportamiento del BIDM en términos de retardo y escalabilidad ha servido de base para la discusión acerca de las dudas que se plantean en el uso de *Blockchain* como tecnología habilitadora de los mercados de datos, así como los retos que deben tenerse en cuenta para que estas utilicen las tecnologías de la *Blockchain* como habilitadores fundamentales en la generación de Espacios de Datos seguros e interoperables.

Abstract

The increasing influence of the Internet of Things over the technological world is undeniable. Both massive data generation and its analysis have proven to be highly useful areas, arousing interest in the research and development of data marketplaces. In this context, in this Master Thesis I have analysed a platform based in Blockchain, so-called Blockchain-based IoT Data Marketplace (BIDM)m over which data producers and data consumers are able to share data in a decentralized, secure and transparent ecosystem. Moreover, in the Thesis I have also carried out a review and analysis of the State of the Art in the topic of data marketplaces in order to put the BIDM in context, as well as to assess its main contributions, and identify the concepts and functionalities that might have recieved little attention in the research of data marketplaces. Afterwards, I have proposed an extension of the BIDM aiming at covering one of the identified gaps, namely the support of asynchronous operation through a subscription-based data exchanging model. Finally, the evaluation that I have carried out of the BIDM behaviour and performance in terms of delay and scalability has been the basis for the discussion that I am presenting on the shortcomings that are typically associated with the use of Blockchain technologies as enablers for data marketplaces. This discussion also includes the evaluation of the challenges that must be taken into account for the creation of secure and interoperable Data Spaces based on Blockchain.

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

BIDM *Blockchain-based IoT Data Darketplace*.

IA Inteligencia Artificial.

IBFT *Istanbul Byzantine Fault Tolerance*.

IoT Internet de las Cosas.

IPC *Inter-process Communications*.

IPFS *InterPlanetary File System*.

P2P *Peer-to-peer*.

PoA *Proof of Authority*.

RNG *Random Number Generator*.

SC *Smart Contract*.

TFM Trabajo Fin de Máster.

URI *Uniform Resource Identifier*.

URL *Uniform Resource Location*.

Capítulo 1

Introducción

En los últimos años, la evolución de Internet ha provocado la aparición de tecnologías y filosofías de funcionamiento cuya implementación no era viable anteriormente. Uno de los ejemplos más conocidos, tanto en su apartado técnico como para el público general, son las plataformas de criptomonedas como *Bitcoin* [1] o *Ethereum* [2]. El auge de estas divisas no solo ha tenido un importante impacto en el ámbito económico, también han abierto las puertas a multitud de aplicaciones que las usan como apoyo para dar un servicio descentralizado. La tecnología en la que se basa este trabajo es *Blockchain*, que es una estructura de datos inmutable que asegura la seguridad de todas las operaciones que se realizan sobre ella. Uno de sus atractivos es su estructura descentralizada: no existe una entidad central que gobierne estas operaciones, sino que son los propios usuarios quienes comprueban la validez de cada una de ellas. Debido a sus características, en la gran mayoría de los casos las transacciones que se realizan sobre una *Blockchain* utilizan criptomonedas como método de pago. Por otro lado, su definición genérica como estructura de datos la hace idónea para el desarrollo de aplicaciones de cualquier tipo.

En paralelo al desarrollo de este paradigma descentralizado empieza a crecer el interés en el manejo de grandes volúmenes de datos, que da lugar a las técnicas de análisis conocidas como *Big Data*. El tratamiento de datos a nivel masivo ha resultado ser de gran interés para aplicaciones como ciudades inteligentes, redes inteligentes o modelos de predicción tanto naturales como sociológicos. Este tipo de sistemas requieren de dos tipos de entidades: productores, que se encargan de la generación u observación de datos; y consumidores, que adquieren estos datos y realizan el procesado correspondiente. Si bien es cierto que en algunas ocasiones estas dos entidades están integradas en el mismo sistema, en muchos casos existe la necesidad de una entidad intermedia o una plataforma que comunique a productores y consumidores. Tradicionalmente, esta funcionalidad la podría llevar a cabo una *Trusted Third Party* (TTP) de manera centralizada. Sin embargo, con el avance de las tecnologías descentralizadas, también es posible realizar esta comunicación de manera automática a través de una *Blockchain*.

Este concepto de un mercado de datos descentralizado basado en contratos inteligentes y *Blockchain* es el objeto de este trabajo. Se va a realizar una implementa-

ción funcional que incluya la integración de una infraestructura IoT de gran tamaño (específicamente *SmartSantander*) como entidad productora de datos, para después analizar su rendimiento como plataforma y optimizar su funcionamiento con el fin de minimizar los tiempos de procesamiento y maximizar la escalabilidad. Asimismo, se van a comparar tanto a nivel cualitativo como cuantitativo diversas filosofías de funcionamiento desde el punto de vista de la compraventa de datos.

1.1 Motivación

Este trabajo utiliza como base una plataforma desarrollada en el TFM de Iván González [3], de nombre BIDM. En ella, se estudia el nicho de mercado de la tecnología Blockchain en el ecosistema de los mercados de datos para la Internet de las Cosas (IoT). Se parte de la idea inicial de crear un sistema donde los proveedores IoT pueden vender las medidas producidas por sus sensores, y aquellos clientes que estén interesados las pueden comprar quedando todos los procedimientos de mercado registrados en la*Blockchain* y, por lo tanto, dando seguridad tanto a unos como a otros. La principal ventaja de este enfoque es la descentralización: no se depende de una TTP para la confirmación de las compras realizadas. Además, el cliente tiene a su disposición toda la información necesaria acerca de la medida que está comprando y quién es el vendedor, dado que solo aquellos sensores autorizados pueden introducir medidas en la Blockchain.

A nivel de implementación, el funcionamiento de la plataforma ya se ha verificado mediante una prueba de concepto. A modo de resumen, se ha desplegado una Blockchain con nodos validadores, un nodo correspondiente al comprador y otro para el proveedor IoT. En esta primera aproximación, es posible insertar una medida en la Blockchain de manera manual a través del proveedor IoT y comprarla, también manualmente, desde un servidor web a través del nodo cliente. Todo ello conlleva el despliegue de contratos inteligentes o SCs que se encargan de realizar las operaciones de manera inmutable y transparente para todos los involucrados. Sobre esta base, el presente TFM busca continuar el desarrollo de la plataforma a través de su análisis exhaustivo e inclusión de nuevas funcionalidades que contrarresten algunos de los problemas que se han identificado mediante dicho análisis.

En primer lugar, se ha visto que uno de los aspectos menos desarrollados de la plataforma es la automatización. Por tanto, una de las mejoras más beneficiosas que se pueden realizar es la automatización de procesos básicos en la *Blockchain*, como pueden ser el registro de nuevas medidas o su compra. Esto va a permitir que el proceso de análisis de rendimiento se pueda realizar a gran escala a través de programas automáticos.

En este momento, la plataforma solo está validada con medidas escritas a mano y totalmente controladas por el desarrollador. Una de las motivaciones más directas es que el mercado de datos funcione con medidas generadas por productores IoT reales, independientemente de su funcionamiento interno. Esto hace evolucionar a la plataforma, que hasta el momento es una prueba de concepto funcional, y la coloca como un

sistema integrable con el mundo real que puede despertar el interés de productores y consumidores reales.

Otra de las motivaciones más relevantes de este trabajo es el análisis del BIDM. Existe una diferencia importante entre algo que funciona y algo que cumple con los indicadores clave de rendimiento (KPI) de manera satisfactoria. Para saber en qué lado del espectro se encuentra la plataforma, es necesario analizar su comportamiento, fundamentalmente, en términos de retardos de procesado y escalabilidad. Este punto conecta directamente con la mejora y optimización. Tras observar los resultados proporcionados por este análisis, el siguiente paso natural es tratar de mejorarlos tanto como sea posible proponiendo nuevas funcionalidades y rediseñando las existentes.

1.2 Objetivos

Con las motivaciones anteriores en mente, se pueden exponer los objetivos del trabajo, que son una materialización de los conceptos comentados. Se listan a continuación las metas específicas que se han planeado para este TFM:

- Estudiar las soluciones de mercados de datos actuales en busca de posibles aportaciones.
- Automatizar los procesos de reinicio de la *Blockchain*, generación configurable de medidas, registro de medidas, y compra de medidas.
- Realizar una integración de la plataforma con proveedores reales de IoT, concretamente *SmartSantander*.
- Mejorar su uso práctico para posibles proveedores y compradores de medidas reales.
- Desarrollar un nuevo modelo funcional que permita a los compradores suscribirse a flujos de datos (o *datastreams*).
- Analizar el sistema desde el punto de vista del uso de recursos computacionales y realizar una comparativa entre los modelos implementados.
- Mejorar y optimizar los modelos funcionales en base a los resultados obtenidos en el análisis.

1.3 Estructura del documento

Tras la introducción realizada en este capítulo, donde se han presentado los conceptos principales, la prueba de concepto que sirve como base del TFM, y las motivaciones y objetivos, el resto del documento sigue la estructura que se describe a continuación. En el Capítulo 2 se presentan con más detalle los conceptos y tecnologías con las que se

ha trabajado, y se ahonda en el estado del arte para evidenciar la novedad y relevancia de este trabajo. En el Capítulo 3 se detalla la implementación del BIDM, explicando su funcionamiento interno y las mejoras desarrolladas. En el Capítulo 4 se presenta un nuevo modelo funcional de la plataforma, que sin sustituir al anterior, añade nuevas funcionalidades y optimizaciones en términos computacionales. El Capítulo 5 contiene una exposición de los análisis y pruebas funcionales realizadas sobre la plataforma, así como comparativas entre diferentes variables y modos de funcionamiento. Por último, en el Capítulo 6 se recapitulan los resultados y las conclusiones obtenidas. Igualmente, se comentan las diferentes líneas de trabajo que surgen de este proyecto.

Capítulo 2

Marco teórico y estado del arte

Con el objetivo de aportar algo de luz acerca de las tecnologías que se han empleado en el desarrollo de este TFM, en este capítulo se van a analizar Blockchain, Ethereum y los modos de funcionamiento que afectan a este trabajo. Posteriormente se van a introducir otras tecnologías complementarias como Go-Ethereum (Geth) o *IPFS*. La última sección sintetiza el estudio del estado del arte que se ha llevado a cabo en el marco del TFM, y una justificación del lugar que ocupa este proyecto dentro de las actuales tendencias e iniciativas de investigación y desarrollo.

2.1 *Blockchain*

La tecnología *Blockchain* se define como una base de datos distribuida que registra bloques de información [4]. Un sistema distribuido consiste en dos o más nodos que colaboran para conseguir un objetivo determinado con un coste individual menor que si se realizara la funcionalidad en un solo nodo. Idealmente, un sistema de este tipo es robusto frente a la desconexión de un subgrupo de los nodos participantes y es capaz de responder a peticiones de manera correcta pese a este impedimento. *Blockchain* sigue esta filosofía además de ser un sistema descentralizado, que no depende de una entidad central gobernadora sino que su funcionamiento y validación dependen de todos los nodos participantes.

Los bloques de información que se registran en la *Blockchain* siguen unas determinadas reglas, que suponen una de las grandes ventajas de esta tecnología:

- Un bloque que se registra en la *Blockchain* es inmutable. Esto significa que una vez se añade, no se puede eliminar ni modificar.

- Todos los bloques que han sido registrados pueden observarse en cualquier momento por todos los usuarios de la *Blockchain*. Los bloques, transacciones y operaciones, así como su contenido, son totalmente transparentes. Esta propiedad no se debe interpretar como un impedimento para realizar operaciones que requie-

ran criptografía, pues el contenido de cada bloque es arbitrario y se puede, por ejemplo, introducir datos cifrados.

- Cada bloque contiene una cabecera que depende de todos los bloques anteriores. Esto asegura que la cadena tiene ciertas propiedades beneficiosas desde el punto de vista de la seguridad.

- Para que un bloque sea introducido en la *Blockchain* de manera definitiva, tiene que ser validado por múltiples nodos (el número exacto de nodos varía de unas cadenas a otras). En general, si un nodo malicioso deseara introducir un bloque ilícito, requeriría de una capacidad de cómputo órdenes de magnitud superior al resto de los nodos participantes.

En la Figura 2.1 se muestra un bloque genérico de la *Blockchain* de Bitcoin [5], incluyendo los campos que lo componen.

La mayoría de los elementos que aparecen en la imagen son comunes a la gran mayoría de *Blockchains*. Se resumen a continuación los más relevantes:

- Las direcciones son identificadores únicos de una cuenta que se utilizan para evitar colisiones en los nombres. Un usuario puede tener múltiples cuentas registradas, de la misma manera que una cuenta puede ser compartida por varios usuarios si así lo desean. Generalmente, las cuentas se desbloquean con una contraseña o con una clave privada.

- La moneda o *token* es la unidad monetaria que se utiliza en la *Blockchain*. Normalmente es una criptomoneda, en el caso anterior es Bitcoin; pero puede tratarse de una divisa ficticia en *Blockchains* privadas.

- Una transacción representa una transferencia u operación realizada sobre la *Blockchain*. Su contenido interno puede ser de cualquier tipo, y dependerá de la funcionalidad de cada plataforma.

- El elemento básico de almacenamiento es el bloque, y como se observa, puede contener una o varias transacciones dependiendo de la velocidad a la que se generen.

- Cabe destacar que, además de las cuentas, los elementos más importantes que tienen interacción directa con la *Blockchain* son los SC. Estos son programas visibles por todos los usuarios y a cuyas funciones se puede acceder en todo momento realizando transacciones hacia su dirección. También es posible activar su funcionalidad automáticamente bajo ciertas condiciones.

Con estos elementos en mente, ya se puede resumir el proceso de registro de nuevos datos en la cadena. Lo que primero se genera es una transacción, ya sea porque un usuario realiza una operación o porque un SC ve activada una de sus funcionalidades.

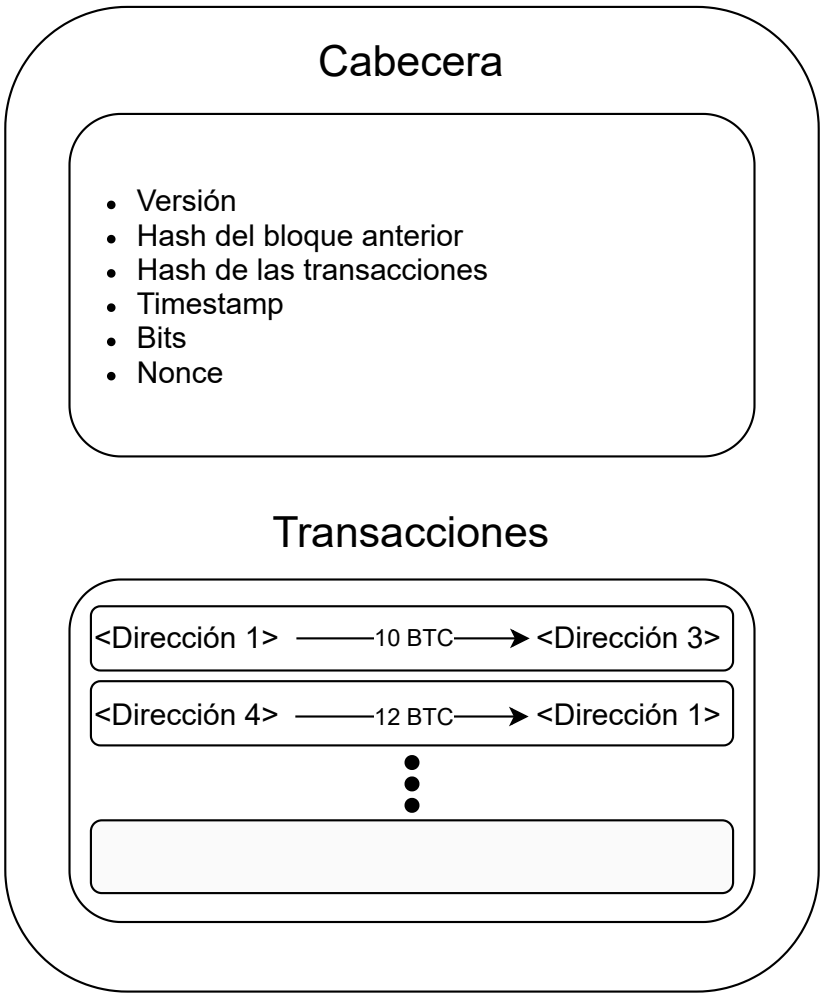


Figura 2.1: Bloque genérico en Bitcoin.

```
1  pragma solidity ^0.4.0;
2
3  contract Contador {
4    uint dato = 0;
5
6    function contar() public {
7      dato = dato + 1;
8    }
9
10   function get() public view returns (uint) {
11     return dato;
12   }
13 }
```

Figura 2.2: Ejemplo de SC.

Esta transacción viaja a través de la red por todos los nodos, y aquellos con autoridad para validar las transacciones deciden si es lícita o no. Una vez suficientes nodos han realizado la validación, la transacción se incluye en un nuevo bloque. Si durante este tiempo se han validado otras transacciones, el bloque contendrá todas ellas. Tras la generación de este nuevo bloque, incluyendo su cabecera, se añade al final de la *Blockchain*. En un apartado posterior se explicará en más detalle el proceso de generación y validación de cada bloque, que requiere de un consenso entre los nodos participantes.

2.1.1 *Ethereum*

En la actualidad existen una gran cantidad de *Blockchains*, y cada una tiene sus peculiaridades y características únicas. Algunas de las más conocidas, o que están experimentando un enorme crecimiento, son *IBM Blockchain*, *Ethereum* o *Stellar*. En este trabajo se va a utilizar una cadena privada Ethereum, ya que tiene múltiples ventajas que la hacen idónea para implementar la funcionalidad deseada. Esta tecnología es de código libre y permite una integración sencilla con los SCs, de tal forma que los desarrolladores pueden crear todo tipo de aplicaciones que funcionan sobre la *Blockchain*. Se muestra en la Figura 2.2 un ejemplo de SC programado en Solidity, el lenguaje utilizado en la gran mayoría de los casos. Este SC, *Contador*, tiene dos funciones. La primera permite aumentar en uno el valor del dato almacenado, que comienza en 0 y está guardado en la *Blockchain*. Con la segunda función se puede leer el dato almacenado en ese momento. Esta sencilla funcionalidad, como en cualquier otro lenguaje de programación, se puede complicar hasta llegar a ejecutar la operativa que desee el desarrollador.

La implementación de SCs no es la única característica particular de *Ethereum*. Por ejemplo, los nodos que participan en la plataforma pueden ser de varios tipos dependiendo de la información que guardan y su funcionalidad. Se comentan sus características a continuación:

- Los **nodos completos** son los pilares básicos ya que realizan las funcionalidades más críticas. Almacenan todo el contenido de la *Blockchain* y su estado actual, además de ser los encargados de recibir las nuevas transacciones y validarlas.

- Los **nodos archivo** están centrados en el almacenamiento. Guardan toda la información que se guarda en los nodos completos, y además cuentan con un histórico de todos los estados en los que ha estado la *Blockchain* desde su origen hasta el momento actual.

- Los **nodos ligeros** guardan información reducida, generalmente las cabeceras de los bloques. Si reciben peticiones de contenido de los bloques, necesitan solicitar esta información a alguno de los otros nodos. A cambio, son nodos que requieren bajos recursos computacionales como procesador, memoria o espacio de almacenamiento.

Como ya se ha comentado, un elemento imprescindible para que los usuarios puedan interactuar con la plataforma son las cuentas. Cada cuenta tiene su propia dirección, que en el caso de *Ethereum* tienen una longitud de 20 bytes o 40 caracteres hexadecimales. Esto supone más de 10^{48} posibles direcciones, un número más que suficiente para evitar colisiones, especialmente si se tiene en cuenta que son direcciones locales de cada *Blockchain*. Las cuentas se dividen en dos tipos; cuentas de usuario y cuentas de SCs. Estas últimas tienen direcciones generadas automáticamente cuando se instancia un SC en la *Blockchain*, y es necesario conocer dicha dirección para poder acceder a las funcionalidades ofrecidas. Por otra parte, las direcciones de las cuentas de usuario se generan en función de la clave pública de quien solicita la cuenta. Una vez creada, para iniciar sesión se utiliza una contraseña que desbloquea el fichero con la clave privada del usuario, que se guarda en un nodo.

Otra de las características de funcionamiento de *Blockchain* es la validación de transacciones. Previamente, se ha introducido este concepto pero no se ha respondido a una pregunta clave: ¿Por qué iba un nodo a realizar trabajo computacional validando una transacción si no recibe nada a cambio? Para solucionar este dilema existe el concepto del **gas**: cada transacción lleva asociado un gas (un número) que el usuario que ha generado la transacción está dispuesto a pagar por su validación y ejecución. Los nodos que realicen estas funciones, algo comúnmente conocido como **minar**, reciben como recompensa este gas en forma de criptomonedas, en este caso *Ether*. También es posible configurar la *Blockchain*, como es el caso de este trabajo, para que las transacciones sean gratuitas y los nodos trabajen por interés del desarrollador.

En *Ethereum* existe un mecanismo adicional que facilita la búsqueda y el filtrado de información, bloques o transacciones anteriores. Los SCs tienen un mecanismo que les permite generar eventos cada vez que se ejecuta alguna de sus funciones. Otros programas externos, usuarios de la *Blockchain* o incluso otros SCs pueden buscar eventos de un determinado tipo, con unos parámetros específicos o que hayan sido generados desde ciertas direcciones, por poner algunos ejemplos. Esto resulta una funcionalidad extre-

madamente útil para buscar determinada información en cadenas muy grandes o muy antiguas, junto a la posibilidad de mantener ficheros de *logs* de manera automática.

2.1.2 Mecanismos de consenso

Los mecanismos de consenso son algoritmos utilizados en *Blockchain* para llegar a acuerdos entre los nodos participantes. Son un elemento necesario en este tipo de plataformas, dado que son descentralizadas y no existe una entidad superior que tome las decisiones. En la mayoría de los casos funcionan por mayoría: se escoge la opción votada por más del 50 % de los nodos, ya que se suelen modelar como encuestas de dos opciones. También existen mecanismos donde es necesaria la aprobación de un determinado porcentaje de los nodos para que la propuesta salga adelante, por ejemplo, un 75 %. Las votaciones deben seguir unas determinadas reglas para ser válidas, que generalmente son de carácter intuitivo. Por ejemplo, una de ellas es que la opción escogida por un nodo debe estar entre las propuestas, y cada nodo puede votar como mucho una vez por cada decisión. Dependiendo de las implementaciones específicas de estos conceptos, existen diferentes mecanismos de consenso que funcionan mejor bajo ciertas condiciones. Se listan a continuación los más relevantes hoy en día:

- ***Proof of Work*** es el mecanismo más extendido, ya que se adoptó inicialmente en *Bitcoin* y se sigue utilizando en muchas de las *Blockchains* más importantes, como la cadena pública de *Ethereum*. Este algoritmo se basa en el minado de los bloques que se añaden a la cadena: los nodos interesados emplean sus recursos computacionales, especialmente CPU (procesador)y GPU (tarjeta gráfica), para resolver puzles criptográficos. El nodo que primero encuentre una solución válida puede introducir el bloque en la cadena y, por tanto, adquirir beneficio económico en forma del gas asociado a las transacciones de ese bloque. Un usuario malicioso que desee introducir bloques inválidos en la cadena debería tener al menos un 50 % de los recursos computacionales de toda la red, aunque realmente existe una componente de suerte debido a la naturaleza aleatoria del minado.

- ***Proof of Stake*** es un mecanismo que nace con la intención de solucionar el problema de los mineros con almacenes llenos de baterías de GPUs. En este caso, el nodo que introduce nuevos bloques en la cadena depende de la cantidad de criptomonedas que tiene cada uno. Así, los usuarios con más dinero invertido en la red tienen una mayor probabilidad de minar los nuevos bloques. Al igual que en el caso anterior, el mecanismo funciona con una componente aleatoria y no por ser un nodo el poseedor de un mayor número de criptomonedas significa que siempre va a ser seleccionado. En muchos casos, se pondera el número de criptomonedas con la antigüedad del usuario. En este sistema se incentiva la "lealtad": los nodos que más tiempo lleven en la red y que más criptomonedas hayan obtenido se ven beneficiados.

- ***Proof of Authority* (PoA)** es el algoritmo más utilizado en las *Blockchains* privadas debido a sus características. Los nuevos bloques solo pueden ser introducidos en la cadena por los nodos que están autorizados para ello. Como esto reduce en gran medida la competitividad entre los usuarios, se pueden optimizar otros aspectos. Por ejemplo, el minado tiende a ser un proceso mucho más rápido ya que se reduce la complejidad de los puzles criptográficos. Además, es posible configurar la *Blockchain* para que se minen nuevos bloques en el momento en que ocurren transacciones. De esta forma, no es necesario que los usuarios esperen al minado de los bloques sino que ocurre al instante. Además, los nodos maliciosos no pueden introducir bloques inválidos mientras no estén autorizados, aunque tengan una capacidad de cómputo superior a los nodos lícitos.

En el caso de este trabajo, el mecanismo de consenso que ofrece más ventajas es PoA. Al tratarse de una *Blockchain* privada, no hay problema en escoger a mano los nodos autorizados para minar. A cambio, las transacciones y los bloques se generan al instante, el uso de recursos computacionales es eficiente y el desarrollador tiene un mayor control para realizar análisis sobre los tiempos de ejecución de cada una de las funcionalidades.

Clique

Dentro de los mecanismos del tipo PoA hay múltiples implementaciones a elegir. Una de las más utilizadas es *Istanbul Byzantine Fault Tolerance* (IBFT), en el cual la inclusión de nuevos bloques en la *Blockchain* pasa por una votación de los nodos autorizados y, además, cada bloque tiene que ser firmado por ellos. Este mecanismo protege de, aproximadamente, un tercio de nodos maliciosos. Como característica adicional, el tiempo que transcurre entre la generación de bloques es constante.

Sin embargo, IBFT no es óptimo para las características de la *Blockchain* que se va a utilizar en este trabajo. En primer lugar, la firma de bloques y la sobrecarga que esto conlleva tanto en tiempo de ejecución como en tamaño de las cabeceras es innecesaria. Además, cuando la *Blockchain* no se esté utilizando, el sistema seguiría generando bloques a intervalos de tiempo constantes, provocando la aparición de bloques vacíos que solo aportan sobrecargas a la plataforma. El mecanismo que se ha decidido utilizar en su lugar es *Clique*, muy común en redes privadas de *Ethereum*. Como primera característica de interés, se permite establecer un gas de 0 para todas las transacciones, que se traduce en que el minado de bloques no aporta recompensa. Adicionalmente se puede eliminar el tiempo entre minado de bloques, de tal forma que no se generan bloques vacíos y las transacciones se introducen en la cadena sin tener que esperar un tiempo constante. El funcionamiento de *Clique* es relativamente estándar: existen nodos validadores y nodos no validadores, y solo los primeros pueden generar y validar nuevos bloques. Para que un bloque se añada a la cadena, más de un 50 % de los nodos validadores tiene que dar el visto bueno. Todos los miembros de la *Blockchain* pueden proponer la promoción de un nodo a validador o la degradación a no validador, pero

son los nodos que en ese momento son validadores los que toman la decisión final. Con este sencillo mecanismo, para que un nodo malicioso se haga con el control de la red la única opción es "convencer" (o piratear) a más del 50 % de los nodos autorizados.

2.1.3 *Go-Ethereum*

A la hora de llevar a la implementación una *Blockchain* de *Ethereum*, hay varias opciones viables donde las tres originales son C++, *Python* y *Go-Ethereum* (Geth) [6]. En este trabajo se ha elegido trabajar con Geth, una implementación escrita en el lenguaje de programación Go y de código libre. Al ser una de las más antiguas, cuenta con soporte para todo tipo de aplicaciones y programas desarrollados en Go. Geth cuenta con un intérprete de comandos con el que se puede interactuar desde cualquier nodo que forme parte de la *Blockchain*, y permite adquirir todo tipo de información acerca de esta. Además, se pueden configurar múltiples parámetros tanto de la plataforma como del nodo en cuestión, realizar conexiones, efectuar nuevas transacciones, y muchas otras funcionalidades. Además, cada nodo cuenta con una interfaz *Inter-process Communications* (IPC) accesible desde cualquier programa corriendo tanto en la misma máquina como fuera. El IPC permite iniciar sesión en una cuenta de la *Blockchain* a través del nodo, y resulta de gran utilidad para realizar funciones automatizadas directamente desde el código. En este trabajo se ha hecho un uso extensivo de esta funcionalidad, que ha permitido mover grandes volúmenes de información y realizar un gran número de transacciones simulando un sistema realista.

2.1.4 IPFS

El último de los elementos principales de la *Blockchain* implementada en este trabajo es la base de datos. Al tratarse de un mercado de medidas, es necesario guardar la información de manera ordenada para que los clientes puedan acceder a los datos que deseen. Una opción más tradicional es usar una base de datos centralizada como MySQL o MongoDB, pero también existen bases de datos descentralizadas que encajan mejor con la naturaleza del resto del BIDM. Dos ejemplos son BigchainDB o IPFS, y la que se ha utilizado es esta segunda [7]. IPFS sigue la filosofía de las redes *Peer-to-peer* (P2P) donde, en lugar de almacenar los datos en un servidor central, se reparten en *chunks* o trozos entre todos los nodos participantes. A la hora de recuperar un dato, el cliente se conecta con múltiples nodos y reconstruye el dato completo. Se muestra en la Figura 2.3 un esquema sencillo que representa esta mecánica.

Algunas de las ventajas que promete IPFS se centran en la eficiencia a la hora de recuperar la información, ya que se descarga desde múltiples nodos simultáneamente. Además, la caída de uno o varios nodos no se traduce en la pérdida total ni temporal de la información. Si bien algunas de estas características no son críticas en esta *Blockchain*, sigue siendo interesante el uso de una tecnología descentralizada y optimizada para este tipo de aplicaciones.

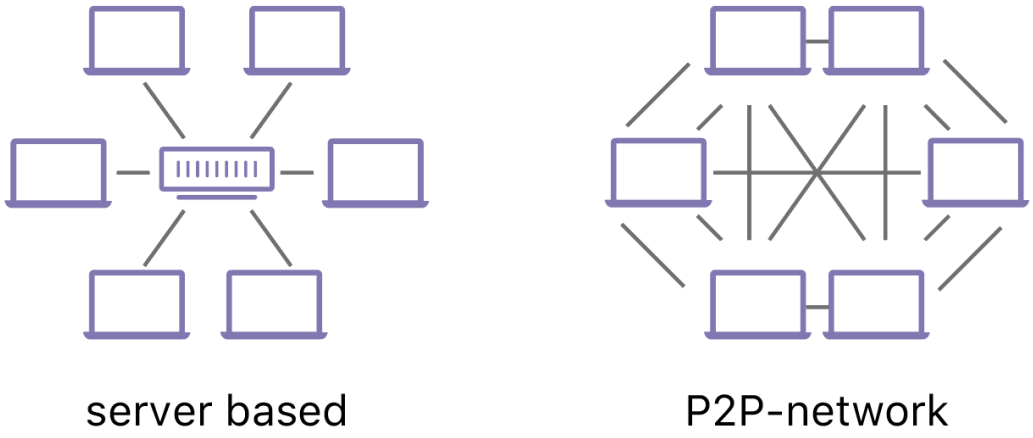


Figura 2.3: Almacenamiento centralizado vs. descentralizado (fuente: [8]).

2.2 Revisión de mercados de datos

El IoT se define [9] como una red basada en protocolos estándar de comunicaciones que hace uso de diversas tecnologías para recolectar y proveer datos que provienen del mundo físico. Como se ve en la Figura 2.4, el número de dispositivos conectados supera ya los 50 mil millones y la cantidad de datos generada es de carácter masivo. Existen muchas funcionalidades que se le pueden dar a estos datos, por ello, no es difícil ver el interés que puede llegar a suscitarse en el ámbito de la compraventa de medidas de sensores, datos privados anónimos, o datos de prueba para el entrenamiento de una Inteligencia Artificial (IA), por poner algunos ejemplos. Estas aplicaciones de una tecnología en auge como es el IoT suponen un atractivo para desarrolladores e investigadores, que comienzan a lanzar ideas y diseños de mercados de datos cuyo fin es interconectar a productores y consumidores a través de una plataforma única.

Las primeras aproximaciones utilizan un enfoque clásico: el mercado de datos es una plataforma centralizada donde existe una entidad gobernadora que se encarga de realizar todas las operaciones necesarias. Por un lado, los productores se registran y ponen a la venta sus datos a un determinado precio. En el otro extremo, los clientes o consumidores acceden a una interfaz que les permite adquirir esos datos. Sin embargo, los procesos de búsqueda, valoración e intercambio de dinero, entre otros, son controlados por la plataforma y las partes involucradas no tienen por qué saber cómo se están llevando a cabo. Es posible realizar una analogía con mercados de productos tradicionales a través de Internet, como pueden ser Amazon o Alibaba. En ellos, es la entidad central la que decide cómo se ordenan los productos en las búsquedas de los

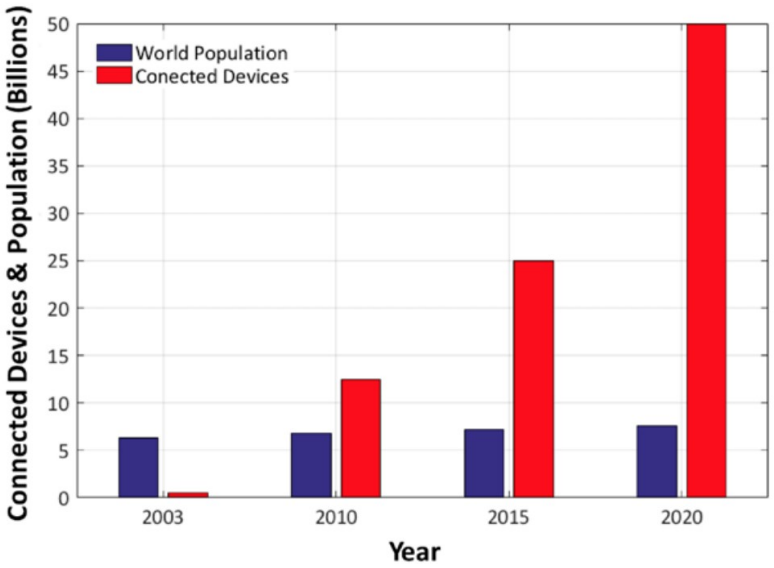


Figura 2.4: Crecimiento del número de dispositivos IoT (fuente: [10]).

clientes, qué porcentaje se lleva la plataforma con cada transacción, o cómo funciona el sistema de valoración. Tanto [11] como [12] son trabajos recientes que optan por esta línea, donde el primero presenta un diseño a nivel conceptual y el segundo introduce un parámetro de *fairness* o "justicia" entre los datos que están a la venta y presenta un modelo matemático de su funcionamiento. Se han realizado múltiples estudios acerca del estado del arte en este ámbito, donde [13] y [14] son algunos de los más significativos. Ambos realizan comparativas entre los documentos publicados al respecto, realizando estudios estadísticos sobre diversas variables como los tipos de productores, la procedencia de los datos o los tipos de monetización. Sin embargo, estos mercados de datos cuentan con algunos retos y barreras que, por su naturaleza, no es posible superar. Se listan a continuación algunos de los problemas de este tipo de soluciones:

- Los compradores pueden estar pagando un precio ligeramente superior al valor del dato, para compensar los gastos de la plataforma.
- Los vendedores no tienen la certeza de que sus datos se estén tratando de forma justa en comparación a otros, tanto en búsquedas como en promociones. ¿Qué datos aparecen en portada?
- Ambas partes necesitan tener confianza absoluta en que la plataforma se va a comportar como debe. Si un comprador reclama que no le ha llegado un dato que ha comprado, es la plataforma quien se debe encargar de solucionar este problema sea quien sea el usuario fraudulento.
- La plataforma tiene acceso a todos los datos que estén a la venta, ya que controla el sistema de pagos. Si la plataforma decide ser deshonesta entonces el mercado

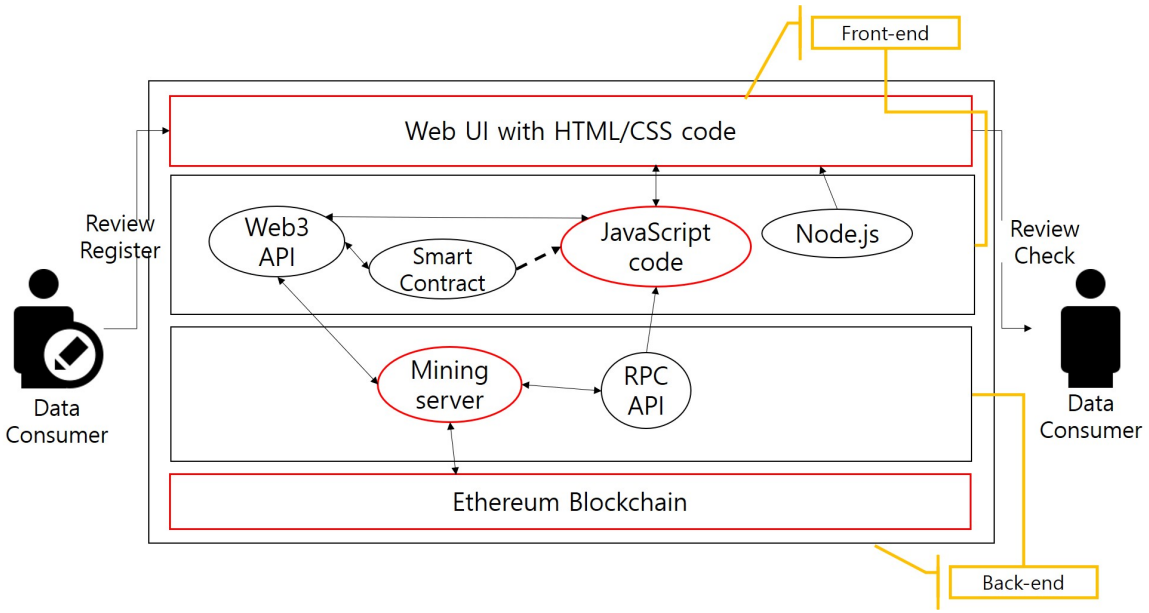


Figura 2.5: Arquitectura del sistema basado en reseñas diseñado en [15] (fuente: [15]).

deja de funcionar, o en el peor caso, sigue funcionando indebidamente sin que los usuarios lo sepan.

Para solventar estos problemas, es necesario un cambio de paradigma que cambie el funcionamiento básico de los mercados de datos. La solución más directa es un modelo **descentralizado**: el mercado no es una entidad central gobernadora, sino una plataforma totalmente transparente en la que todas las operaciones se realizan de manera conocida y los participantes colaboran entre sí para llegar a acuerdos sobre su funcionamiento. Siguiendo esta filosofía aparecen los mercados de datos sobre *Blockchain*, que han despertado un interés creciente con los años en la comunidad. Algunos ejemplos son [17], [18], [19] o [20], que se centran en las ideas principales y realizan un diseño conceptual de la plataforma. Otros trabajos se especializan en mercados de datos IoT, como [21], [22], [23] o [24], aunque la mayoría de los existentes no entran a la implementación de los diseños presentados. Sin embargo, hay multitud de trabajos que presentan ideas novedosas o interesantes, como [25], que centra su diseño en la compraventa de datos privados anónimos cumpliendo con la reciente norma GDPR. [15] y [26] idean sendos sistemas de reseñas para que los compradores valoren las medidas que compran, o que comprador y vendedor se pongan nota el uno al otro. Se muestra en la Figura 2.5 la arquitectura de uno de estos sistemas. El uso que los compradores le dan a los datos también es una variable que se tiene en cuenta en muchos documentos. Por ejemplo, [16] plantea un mercado cuyo objetivo es la distribución de datos masivos para *machine learning* utilizando entornos de ejecución de confianza. Su estructura se puede observar en la Figura 2.6. Por su parte, [27] tiene como fin la compraventa de datos médicos de pacientes anónimos. Otros trabajos buscan un modelo de monetización que

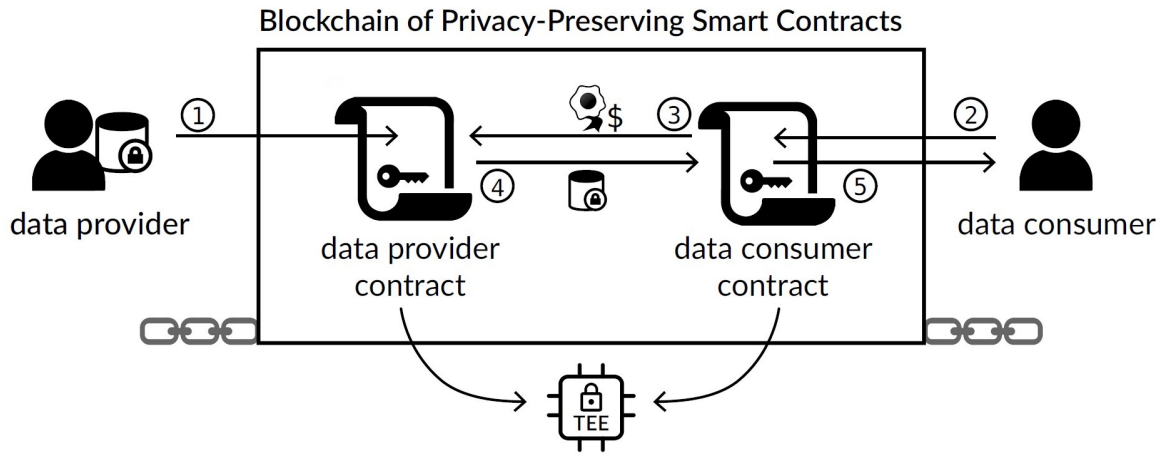


Figura 2.6: Estructura de los entornos de ejecución de confianza empleada en [16] (fuente: [16]).

pueda atraer a los clientes a utilizar su plataforma, como [28] o [29].

A pesar de existir un gran número de diseños de mercados de datos sobre *Blockchain*, donde los mencionados son solo una pequeña parte, hay un aspecto que es común a la gran mayoría. Estos trabajos se centran en un diseño detallado de sus plataformas y, en algunos casos, estimaciones de su funcionamiento. Sin embargo, limitados son los casos en que se realiza una implementación del diseño planteado. Además, los resultados que muestran estos trabajos giran alrededor del sistema de gas utilizado en *Blockchain*, estando por tanto más centrados en la economía que en el funcionamiento básico. Teniendo esto en cuenta, se plantean unas propuestas de valor que convierten este trabajo en un avance significativo a la hora de analizar la viabilidad y el rendimiento de este tipo de soluciones. Se listan a continuación dichos aportes, directamente relacionados con la motivación de este trabajo:

- La implementación completa del mercado de datos, demostrando el correcto funcionamiento a nivel práctico del diseño realizado.
- El análisis de rendimiento del BIDM en cuanto al uso de recursos computacionales, tanto en tiempo como en espacio de disco.
- La comparativa entre diferentes modos de funcionamiento, tanto a nivel cualitativo como cuantitativo.
- La integración con proveedores de IoT reales, sea cual sea su funcionamiento interno.

Para mayor claridad a la hora de comparar las referencias que se han analizado en esta sección, así como el BIDM, se ha realizado una tabla en la que se recogen las características principales de cada uno de los mercados de datos. El principal objetivo es sintetizar los elementos más importantes para poder ver de un vistazo cuáles de ellos cubre cada uno de los artículos. La Tabla 2.1 muestra los resultados de esta comparativa. Las columnas representan lo siguiente: "*Blockchain*" indica si el mercado de datos funciona sobre la tecnología *Blockchain* o no; "Implementación" muestra si la plataforma diseñada está implementada o se queda en un concepto; "Análisis (gas)" manifiesta si el trabajo realiza un análisis del gas consumido por sus transacciones, "Análisis (rend.)" dictamina si el artículo analiza el rendimiento computacional del diseño realizado de manera extensiva, ya sea un análisis teórico o una demostración práctica; "Uso general" indica si los datos a comprar y vender pueden ser de cualquier naturaleza o están limitados a una aplicación específica; y "Monet." señala si el artículo ha descrito o implementado un modelo de negocio para monetizar su plataforma.

Tabla 2.1: Comparativa de artículos sobre mercados de datos.

Nombre del artículo (corto)	*Blockchain*	Implementación	Análisis (gas)	Análisis (rend.)	Uso general	Monet.
Intelligent DM [11]	✗	✗	✗	✗	✓	✗
Algoritmic Solution [12]	✗	✗	✗	✓	✓	✗
SC-based review system [15]	✓	✓	✓	✗	✓	✗
Sterling [16]	✓	✓	✗	✗	✗	✗
Decentralized IoT [17]	✓	✗	✗	✗	✓	✗
Blockchain based [18]	✓	✗	✗	✗	✓	✗
Blockchain enabled [19]	✓	✗	✗	✗	✓	✗
Blockchain technology [20]	✓	✗	✗	✗	✗	✗
Blockchains and SCs [21]	✓	✗	✗	✗	✓	✗
Data MP for IoT [22]	✓	✓	✓	✗	✓	✗
Towards a Blockchain [23]	✓	✗	✗	✗	✓	✗
Towards a decentralized [24]	✓	✓	✗	✗	✓	✗
Digital Me [25]	✓	✗	✗	✗	✗	✓
BlendSM-DDM [26]	✓	✗	✗	✗	✓	✗
Blockchain-based medical [27]	✓	✓	✓	✗	✗	✗
Agora [28]	✓	✓	✓	✓	✓	✗
Monetization [29]	✓	✓	✗	✗	✓	✓
BIDM	✓	✓	✗	✓	✓	✗

Capítulo 3

Modelado y funcionamiento del BIDM

El mercado de datos analizado en este TFM se compone de diversos componentes interconectados entre sí, cuyo objetivo es la comunicación transparente entre productores de todo tipo y consumidores. En este capítulo se va a detallar la estructura de la plataforma, explicar el funcionamiento de cada componente y el mecanismo detrás de las operaciones más relevantes, como los registros y las compras de medidas.

3.1 Arquitectura del BIDM

El núcleo del BIDM es una *Blockchain Ethereum*, que registra la información clave acerca de las nuevas medidas registradas por productores y adquiridas por clientes, y otros aspectos como la creación de nuevas cuentas o el intercambio de *tokens* de compra. En torno a esta estructura básica giran el resto de componentes, que se detallan en la Figura 3.1.

La *Blockchain* tiene como participantes los nodos interconectados entre sí. Cada comprador puede crear una cuenta *Ethereum* en uno de estos nodos, protegida por contraseña, y acceder a todas sus funcionalidades. El color verde se corresponde con todo lo que pertenece a la *Blockchain*, que además de los nodos, son los SCs. Como ya se ha comentado, son ficheros de código que se instancian en la *Blockchain* y a cuya funcionalidad se puede acceder conociendo la dirección *Ethereum* de la instancia. Además, el código es libre y todo usuario puede leer su contenido y saber exactamente cómo se van a llevar a cabo las operaciones. En color naranja se observan los tres SCs, que están divididos según su papel en la plataforma: datos, acceso y balance. El contenido de estos contratos se detallará más adelante.

Por otra parte, en color azul aparecen los usuarios de la *Blockchain*, tanto un productor (o vendedor) como un consumidor (o comprador). Los productores tienen la posibilidad de utilizar un *Middleware* cuya función es adaptar su modelo de medidas al formato que acepta el BIDM. Los consumidores, por su parte, tienen a su disposición

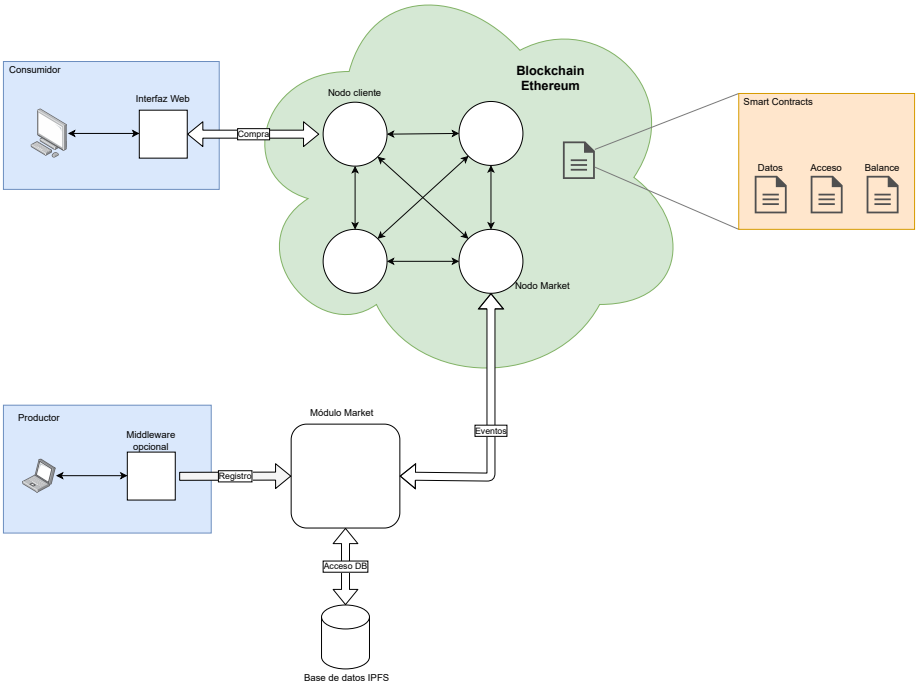


Figura 3.1: Estructura del mercado de datos.

una interfaz web para interactuar con la *Blockchain*: realizar compras, ver las medidas que han adquirido o el saldo que tienen en la cuenta, por mencionar algunos ejemplos. Esta interfaz web tiene acceso directo a un nodo de la *Blockchain* que se encarga de gestionar las cuentas de cliente.

Los últimos componentes, que se representan sin color por ser elementos "neutros", son el módulo *Market* y la base de datos IPFS. Este módulo tiene acceso a una cuenta de administrador en un nodo, y le permite realizar operaciones especiales en los contratos. Se encarga, entre otras cosas, de recibir peticiones de los productores para añadir nuevas medidas al mercado, completar los procesos de compra de los consumidores, o añadir nuevas cuentas al BIDM. Además, tiene acceso a la base de datos tanto para guardar las medidas nuevas como para recuperarlas cuando se produce una compra.

3.1.1 *Smart Contracts*

Los contratos inteligentes que realizan operaciones sobre la *Blockchain* a petición de los usuarios se instancian al inicio, generando una dirección que se hace pública para que se pueda acceder a ellos de manera sencilla. Tanto la interfaz web como el módulo *Market* conocen estas direcciones y pueden interactuar con los SCs de forma automática, de tal forma que los usuarios no necesitan grandes conocimientos sobre la tecnología que utiliza la plataforma para poder hacer uso de ella. Como se ha comentado previamente, existen tres SCs que se reparten la funcionalidad.

```
1    function storeInfo(bytes32 hash, string memory uri, string[]
        memory topic) public {}
2    function deleteMeasurement(bytes32 hash) public {}
3    function getIoTAddress(bytes32 hash) public view returns (address
        ) {}
```

Figura 3.2: Funciones más significativas del SC datos.

```
1    function addAccountToRegister(address producerAddr, string memory
         producerName) public {}
2    function removeAccountFromRegister(address producerAddr) public
        {}
3    function addPubKey(string memory pubKey) public {}
```

Figura 3.3: Funciones más significativas del SC acceso.

SC datos

El SC datos tiene como objetivo principal la gestión de la información dentro de la *Blockchain*. Se muestra en la Figura 3.2 un resumen con las primitivas de algunas de las funciones que contiene. La función *storeInfo* es posiblemente la más relevante, ya que permite registrar una nueva medida en la *Blockchain* dado su *hash*, la *Uniform Resource Identifier* (URI) en la que se encuentra dentro de la base de datos, y una descripción para lectura de los consumidores previa compra. Este último parámetro se ha optimizado para reducir espacio en disco, ya que previamente se trataba de un *string* de longitud variable. Ahora, se adopta un formato JSON haciendo que la descripción sea fácilmente interpretable por otros programas. Al contrario que *storeInfo*, la función *deleteMeasurement* permite al administrador eliminar una medida que pueda estar corrupta o ser fraudulenta. Por otra parte, la función *getIoTAddress* devuelve a quien la llame la dirección *Ethereum* de la cuenta que registró una determinada medida. Este contrato posee más funciones que facilitan tanto a los usuarios como al administrador la interacción con la *Blockchain*.

SC acceso

El SC acceso es el encargado de gestionar las cuentas *Ethereum* de los clientes. En la Figura 3.3 se muestran una vez más las primitivas de las funciones más relevantes contenidas en este contrato. Tanto *addAccountToRegister* como *removeAccountFromRegister* son funciones que permiten a los productores registrar o eliminar sus sensores de la lista blanca de la *Blockchain*. Solo aquellas medidas que proceden de direcciones en esta lista serán registradas y vendidas en la plataforma. Estas funciones solo responden ante llamadas del administrador, para evitar que cualquiera pueda registrar productores fraudulentos o eliminar el acceso de productores lícitos. Por otra parte,

```
1     function getPriceMeasurement(bytes32 hash) public view returns(
          uint256) {}
2     function purchaseMeasurement(bytes32 hash) public {}
3     function completePurchase(bytes32 hash, address buyer, bytes32
          txHash) public {}
4     function sendTokenToClient(address to, uint256 total) public {}
```

Figura 3.4: Funciones más significativas del SC balance.

addPubKey registra en una lista visible por todos los participantes la clave pública de aquellos que realicen operaciones en la *Blockchain*.

SC balance

En el SC balance se tramita todo lo relacionado con las compras y las transacciones de *tokens*. La Figura 3.4 muestra las primitivas de algunas de las funciones más importantes que contiene. Los consumidores que desean adquirir una medida realizan una llamada a *purchaseMeasurement*, que registra su petición en la *Blockchain* de manera inmutable. Posteriormente, este evento es escuchado por el módulo *Market* y, desde la cuenta de administrador, se completa la transacción con la función *completePurchase*. Si un usuario, ya sea comprador o vendedor, desea ver el precio de una medida, lo puede hacer a través de la función *getPriceMeasurement*. Además, el administrador puede añadir *tokens* sin límite a las cuentas de los usuarios mediante *sendTokenToClient*. Este contrato, además, incluye como librería varias funciones estándar de transferencia de *tokens* entre cuentas, permisos para realizar operaciones en nombre de otro usuario, o acceso público a información básica sobre la *Blockchain*.

3.1.2 Componentes del BIDM

En esta sección se va a explicar en mayor detalle la funcionalidad de cada uno de los componentes que forman parte de la *Blockchain*, además de los SCs. A nivel de implementación, el módulo *Market* está dividido en dos programas *Go-Ethereum* y la interfaz web está realizada en *Javascript* con el módulo *node* [30]. Además, se han realizado varios programas en *Python* para automatizar tareas de registro, compra e interpretación de *logs*.

Market core

El *core* es uno de los programas que componen el módulo *Market* visto al inicio de esta sección, y aquel que tiene acceso al nodo de administrador. Por consiguiente, este programa tiene permisos para realizar cualquier tipo de operación sobre la *Blockchain*. En la Figura 3.5 se pueden apreciar las primitivas de varias de las funciones relevantes de este componente.

```
1    func initialize() (localClient, localClient) {}
2    func ManagePurchases(ethClient ComponentConfig) {}
3    func completePurchase(ethClient ComponentConfig, vLog types.Log)
        {}
4    func autoBuy(ethBuyer libs.ComponentConfig) {}
```

Figura 3.5: Funciones más significativas del *Market core*.

```
1    func (myLocalClient localClient) EventListener(w http.
        ResponseWriter, req *http.Request) {}
2    func ProcessMeasurement(ethClient ComponentConfig, body map[
        string]interface{}) error {}
3    func insertDataInBlockchain(ethClient ComponentConfig, dataStruct
         DataBlockchain) error {}
4    func AddToIPFS(ipfs icore.CoreAPI, file *bytes.Reader) (string,
        error) {}
```

Figura 3.6: Funciones más significativas del *IoT proxy*.

En primer lugar, *initialize* es llamada al ejecutar el *Market* y se encarga de realizar las preparaciones necesarias para habilitar la comunicación con la *Blockchain* y los SCs. Para ello, a partir de un fichero de configuración, rellena los campos de la estructura que funciona como cliente *Ethereum* y se utiliza para hacer llamadas a la *Blockchain*. Además, esta función informa a los SCs de las direcciones en las que están el resto de SCs y proporciona a la cuenta de administrador *tokens* para realizar las transacciones que desee. *ManagePurchases* crea un canal *Go* que se encarga de escuchar los eventos que se generan en la *Blockchain*, y cuando detecta una petición de compra, llama en paralelo a la función *completePurchase*. Esta se comunica con el SC balance para completar la compra en nombre del administrador de la plataforma, tras comprobar que todos los parámetros involucrados en la petición de compra son correctos. Por último, *autoBuy* es una función de automatización pensada para que el desarrollador pueda realizar compras en masa, con el fin de llevar a cabo los análisis y medidas de este trabajo de manera cómoda y eficiente. Tanto esta función como otras menos significativas contenidas en este componente se han incluido como mejora respecto al modelo inicial.

IoT proxy

El *IoT proxy* es el segundo fragmento del módulo *Market*, que tiene interacción directa con los productores de medidas y escucha sus peticiones. La separación en dos programas se justifica con la posibilidad de ejecutar cada uno en una máquina diferente, sin que esto interfiera en su funcionalidad. Se observan en la Figura 3.6 algunas de las funciones que forman parte de este componente.

EventListener es una interfaz REST que espera peticiones del tipo HTTP POST por parte de los productores. En el cuerpo de las peticiones se envía la medida que el productor desea registrar en la *Blockchain*, y que lleva información relevante como el sello de tiempo (o *timestamp*) en el que se ha generado. Tras recibir una medida, se comprueba que quien la ha enviado tiene acceso a la plataforma como productor y se llama a *ProcessMeasurement*. En esta, se firma la medida con la clave del administrador y se extrae la información que se va a publicar en la *Blockchain*. Esta información está compuesta por el *hash* de la medida, varios parámetros que conforman la descripción y la *Uniform Resource Location* (URL) a la que se ha de acceder para visualizarla tras una compra. Esta URL es generada por la base de datos IPFS tras la ejecución de la función *AddToIPFS*, que guarda la medida encriptada en dicha localización. Tras el procesado de esta información, se llama a la función *insertDataInBlockchain* para finalizar el proceso de registro. En esta última, se realizan llamadas al SC datos para guardar la información de la medida y al SC balance para registrar su precio de venta. Este componente ha recibido varias optimizaciones de rendimiento respecto a su implementación anterior. Dos de las más importantes consisten en la eliminación de dos bucles que realizaban operaciones redundantes de comprobación, cuya función era asegurar que tanto las medidas como su precio se habían guardado correctamente en los SCs. Durante la realización de este trabajo, se ha comprobado experimentalmente que las llamadas a SCs a través del paquete *ethclient* de Go ya hacen estas comprobaciones automáticamente y solo devuelven el control al programa llamador una vez se ha confirmado su correcta ejecución. Por tanto, la supresión de los bucles mencionados elimina sobrecarga adicional en el programa y acelera su rendimiento.

3.1.3 Utilidades adicionales

Los siguientes componentes no forman parte intrínseca del BIDM, pero son fundamentales para el cumplimiento de los objetivos del proyecto en tanto en cuanto facilitan la validación y evaluación del comportamiento y rendimiento del BIDM. En el TFM se ha llevado a cabo el diseño, desarrollo y/o mejora de todos ellos.

Interfaz web

El elemento que permite a los compradores acceder a todas las funcionalidades de la *Blockchain* sin necesidad de tener grandes conocimientos técnicos de *Ethereum* es la interfaz web. Este módulo tiene acceso a un nodo de cliente de la *Blockchain*, lo que le permite interactuar con los SCs. La estructura de la página web está detallada en la Figura 3.7.

En este esquema, el color verde representa las URLs accesibles por cualquier usuario, es decir, son públicas. En color naranja, por otro lado, están las URLs privadas o que solo son accesibles por usuarios autenticados. La primera ruta que se muestra al usuario es el índice, en el cual se muestra información básica relativa a los últimos bloques minados en la cadena y a las últimas transferencias que involucran el intercambio

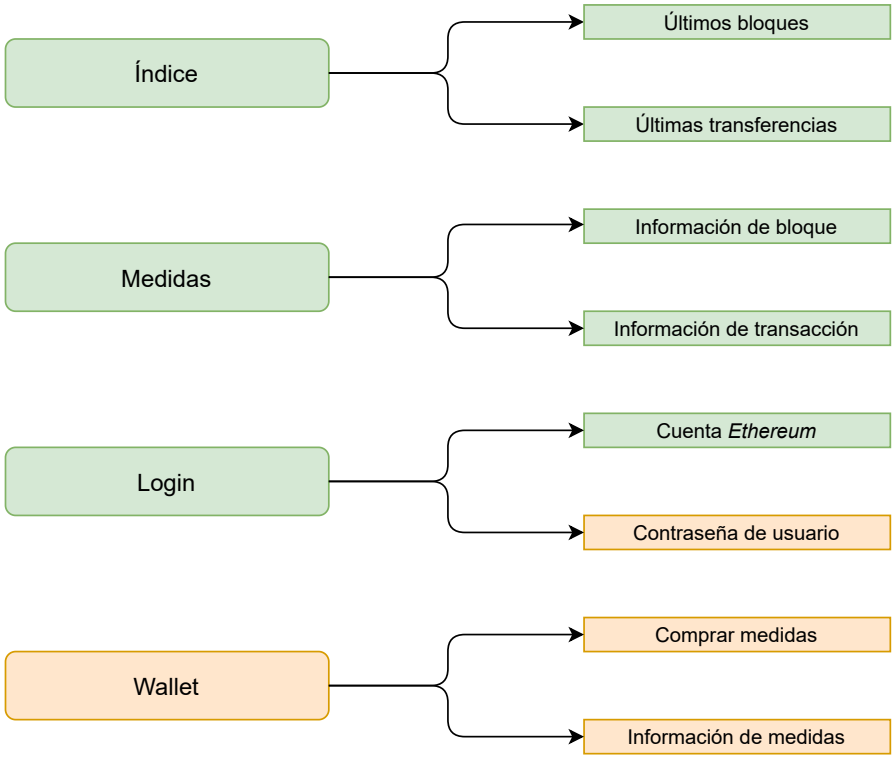


Figura 3.7: Estructura de la interfaz web.

de *tokens* del mercado. Además, se incluyen referencias al resto de rutas accesibles por el usuario. En la ruta "Medidas" se muestra un listado de las últimas medidas registradas por los productores, detallando su precio, *hash* y descripción para que el cliente pueda decidir si está o no interesado en su compra. Además, es posible acceder a la metainformación de cada bloque y cada transacción de la *Blockchain* para ver datos como las direcciones *Ethereum* involucradas o el sello de tiempo. La ruta de inicio de sesión permite a los compradores insertar su dirección *Ethereum*, pública, y su contraseña personal, privada, para acceder a funciones adicionales. Además de aspectos básicos como cerrar sesión o ver en todo momento el número de *tokens* de los que dispone la cuenta, se abre el acceso a la ruta "Wallet". Desde aquí, los usuarios pueden comprar nuevas medidas, listar aquellas que ya han comprado, y acceder a su información. Por supuesto, esto incluye el valor de la medida y su sello de tiempo, que es el interés principal de los compradores.

La interfaz web ha sido objeto de numerosas mejoras para minimizar el retardo que sufrían los clientes al cargar las diferentes páginas. Uno de los problemas más comunes era que se trataba de mostrar todas las medidas que pasaran los filtros. Por ejemplo, en la ruta "Medidas" se mostraban todas las medidas disponibles para su compra. En la prueba de concepto esto no suponía ningún problema, pero durante el desarrollo de este trabajo se han llegado a tener decenas de miles de medidas disponibles. Las páginas comenzaban a tener tiempos de carga de varios segundos, hasta el punto en el

que tenía lugar un *timeout* y el servidor web se caía. Estos problemas, entre otros, se han solucionado limitando el número de elementos que se muestran simultáneamente en cualquiera de las rutas.

Programas adicionales

En conjunto con los componentes que se han ido explicando en esta sección, se han desarrollado una serie de programas y *scripts* para facilitar el análisis de la plataforma. Se listan a continuación algunos de los más relevantes:

- Un **generador** de medidas sintético parametrizable en *Python*, capaz de generar peticiones con diferentes distribuciones (Poisson, regular) variando la media o el tiempo entre generaciones.
- Un **servidor** *Python* que actúa como *middleware* para los vendedores IoT. Este programa recibe las medidas de productores reales como *Smart Santander* y las formatea para que se puedan procesar directamente en el componente *IoT proxy*.
- Un programa de **reset** de la *Blockchain*, escrito en *bash*, que reorganiza los ficheros de configuración y almacenamiento para reiniciar la *Blockchain* en un comando y evitar que las pruebas de rendimiento desborden la capacidad de las máquinas virtuales.
- Un sistema de **logs** que se escriben desde los diferentes componentes que forman parte de la plataforma, registrando los sellos de tiempo de cada procedimiento para analizar posibles cuellos de botella en las operaciones más pesadas computacionalmente.

3.2 Operaciones fundamentales

El funcionamiento de la plataforma, desde el punto de vista del usuario, se basa fundamentalmente en la ejecución de operaciones que modifican el estado del mercado de datos. Teniendo en cuenta que los usuarios pueden ser productores o consumidores, las operaciones básicas son tanto el registro de nuevas medidas por parte de los primeros como la compra por parte de los segundos. Ambas requieren la comunicación y cooperación entre los elementos vistos previamente en este capítulo, y la ejecución de múltiples funciones tanto en los componentes como en los SCs. En esta sección se va a presentar un análisis pormenorizado de estas operaciones, detallando las funcionalidades que realiza cada componente y las comunicaciones entre ellos.

3.2.1 Registro de medidas

La primera operación clave es el registro de medidas, que se lleva a cabo por parte de los productores IoT. Un esquema del flujo de ejecución de esta operación se puede observar

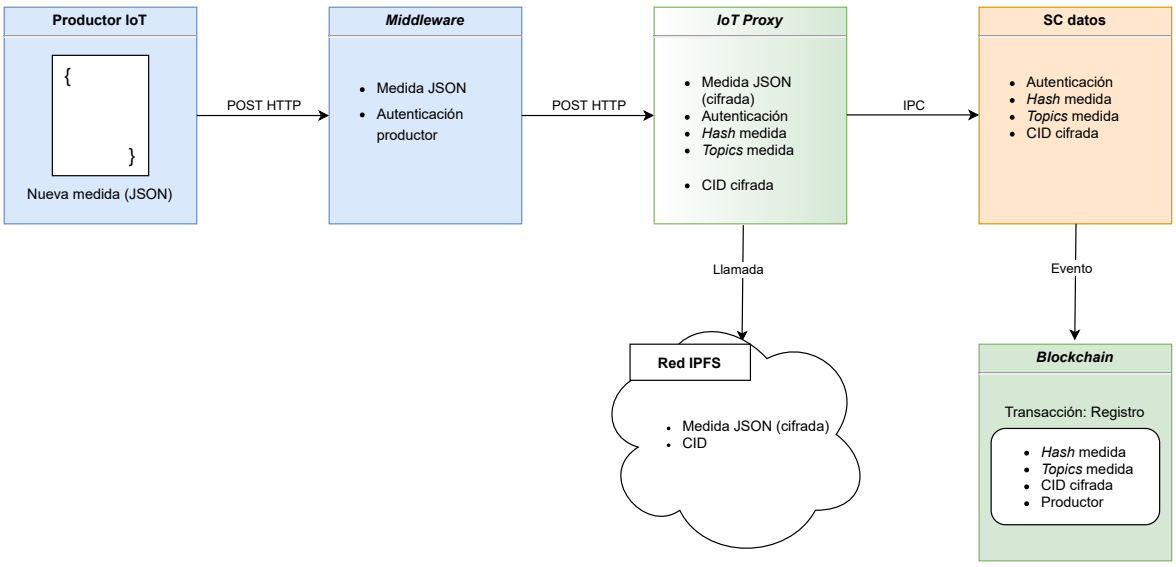


Figura 3.8: Esquema de flujo del registro de medidas.

en la Figura 3.8. El código de colores es similar al explicado en la sección anterior. Los elementos azules están del lado del usuario y no tienen acceso a la *Blockchain*. El color blanco representa los elementos "neutros", el módulo *Market* y la base de datos. En verde aparecen los componentes que pertenecen a la *Blockchain* o tienen acceso a ella a través de uno de los nodos *Ethereum*, y en naranja, los SCs. Los elementos que tienen un gradiente de color poseen las funcionalidades representadas por ambos colores. Se explican a continuación cada una de las fases del proceso de registro de medidas:

1. El productor IoT, o cualquiera de sus sensores, genera una nueva medida y desea registrarla en la *Blockchain*. Para ello, envía dicha medida en formato JSON al componente que actúa de *Middleware* entre cada productor y el mercado, o si no lo necesita, se comunica directamente con este último. Esta comunicación se realiza con una petición HTTP POST donde el cuerpo lleva la medida en cuestión.

2. El *Middleware* se encarga de recibir la medida en formato JSON y formatearla correctamente en caso de ser necesario para que pueda ser aceptada por la plataforma. Además, extrae los datos de autenticación del productor y se lo envía todo al *IoT proxy* con una petición HTTP POST.

3. El componente *IoT proxy*, que forma parte del módulo *Market* y por tanto tiene acceso a un nodo de la *Blockchain*, recibe la medida y los datos de autenticación del productor. En este elemento se extrae información de la medida, como los *topics*, y se realiza el *hash* de la misma. Estos *topics* son los datos de interés de los que se extrae la descripción, pero en la sección posterior se va a explicar un uso mucho más extensivo de estos. Este componente hace una llamada a la red IPFS para guardarla después de cifrarla para restringir su acceso, y en la

respuesta recibe un CID, que es un identificador para poder acceder a la medida en el futuro.

4. A continuación, el *IoT proxy* se conecta al nodo *Market* a través del IPC y llama al SC datos para registrar la información de la medida en la *Blockchain*. Para ello, se utiliza la función *storeInfo* del contrato, que requiere los datos de autenticación del productor, el *hash* de la medida, sus *topics* o descripción, y la CID en la que se encuentra. Esta CID ha sido previamente cifrada para evitar que se convierta en información pública.

5. En el SC datos se ejecuta la función mencionada, que genera un evento en la *Blockchain*. Esto completa la transacción de registro, y en la *Blockchain* se guarda la información básica necesaria para poder comprar esta medida de manera segura: la dirección *Ethereum* del productor, el *hash* de la medida como identificador, la CID para acceder a ella en la base de datos y los *topics* o descripción para que los compradores decidan si les interesa o no. Como la CID está cifrada, los usuarios no pueden acceder a ella aunque la información de la transacción sea pública y es necesaria la operación de compra para poder adquirir una medida.

3.2.2 Compra de medidas

La siguiente operación básica es la compra de medidas, iniciada por los consumidores. El esquema del flujo de ejecución de esta operación se muestra en la Figura 3.9. El código de colores es equivalente al que se ha comentado para la operación de registro. Se explican a continuación cada una de las fases del proceso de compra de medidas, que genera varias transacciones:

1. Un comprador, desde la interfaz web, observa las medidas registradas y decide que desea comprar una de ellas. Para esto, realiza una petición desde *JavaScript* que incluye el *hash* de la medida a comprar como identificador. Esta petición pasa, mediante una llamada, al núcleo del servidor web.

2. El *NodeJS*, que se comunica tanto con la interfaz web como con la *Blockchain* a través de un nodo de cliente, recibe la petición. Extrae la información de la autenticación de cliente, que tiene que estar registrado para haber realizado la compra, y el *hash* de la medida a comprar. A través de la interfaz IPC, este componente realiza una llamada al SC balance.

3. En este contrato se ejecuta la función *purchaseMeasurement*, que genera una transferencia de *tokens* intermedia desde el comprador hacia una cuenta *dummy* controlada por el administrador. Esto genera un evento en la *Blockchain*.

4. En esta se registra una transacción de compra, que incluye el *hash* de la medida comprada y las direcciones de vendedor y comprador. Además, la generación de

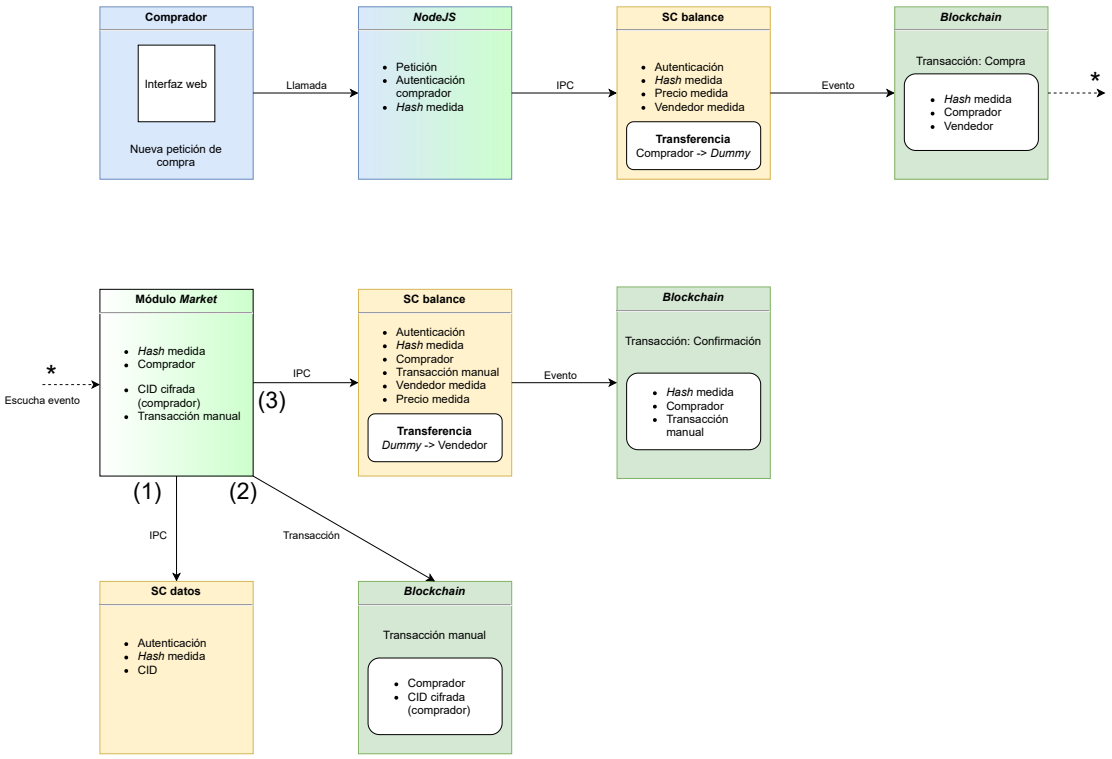


Figura 3.9: Esquema de flujo de la compra de medidas.

este nuevo evento permite a programas con acceso a la *Blockchain* escucharlo y activar alguna de sus funciones en consecuencia.

5. En este caso, es el módulo *Market* quien se encuentra a la escucha de estos eventos de compra. Cuando se genera uno, este componente extrae el *hash* de la medida comprada y la identidad del comprador. Con esta información, realiza una llamada a través del IPC del nodo *Market* al SC datos, más concretamente a la lista de medidas y los campos que se registraron en su momento. Con la autenticación adecuada por tratarse de un nodo de administrador, el *Market* puede obtener la CID de la base de datos en la que se encuentra la medida.

6. A continuación, el *Market* envía una transacción manual a la *Blockchain* (es decir, que la genera directamente en lugar de utilizar un contrato). En ella se registran en la *Blockchain* la identidad del comprador y la CID cifrada, pero esta vez con la clave pública del comprador, para que solo este pueda acceder a la medida en la base de datos. La generación de esta transacción devuelve un *hash* de transacción.

7. Por último, el *Market* accede al SC balance a través del IPC, llamando a *completePurchase*. Los argumentos de esta llamada son la autenticación de administrador, el *hash* de la medida comprada, la dirección del comprador y el *hash* de

la transacción generada anteriormente. En el SC se realiza la segunda parte de la transferencia de *tokens*, esta vez desde la cuenta *dummy* hacia el vendedor. Si algo sale mal durante el proceso, la compra se revoca y el *dummy* le devuelve los *tokens* al comprador.

8. Finalmente, se genera un evento que provoca una nueva transacción en la *Blockchain*, del tipo confirmación de compra. La información registrada incluye el *hash* de la medida, la dirección del comprador y el *hash* de la transacción manual anterior en la que el comprador puede reclamar el acceso a su nueva medida.

Capítulo 4

Extensión del BIDM para soporte de *datastreams*

En el capítulo anterior se ha cubierto el modo de funcionamiento tradicional de compras de medidas, en el que los consumidores seleccionan cada medida individual que desean comprar. En este capítulo se presenta la extensión de la plataforma que se ha desarrollado para agregar una nueva funcionalidad: las suscripciones a flujos de datos o *datastreams*. El objetivo es aumentar la flexibilidad del BIDM, soportando un nuevo modelo de funcionamiento que permite a los consumidores seleccionar la operación que les resulte más conveniente. En primer lugar se describe el diseño que se ha hecho para soportar la suscripción a flujos de medidas detallando el proceso por el que se realiza una de dichas suscripciones. A continuación se detallan los cambios y mejoras realizados en los componentes descritos en el Capítulo 3. La comparativa entre ambos modos de funcionamiento se analiza con mayor detalle en el Capítulo 5.

4.1 Suscripciones

En el Capítulo 3, se pudo observar que el proceso de compra es complejo y requiere la intervención de muchos de los componentes que forman parte del BIDM. Además, cada compra genera tres transacciones en la *Blockchain*, lo cual puede llegar a ser una sobrecarga a medida que crecen el número de medidas y de compradores. Como alternativa, se ha propuesto un modelo de *topics* y suscripciones. Cada medida registrada forma parte de unos determinados *datastreams*, a los que se les asigna un *topic* descriptivo. Por ejemplo, es una medida generada por el sensor X (*topic* 1), muestra datos de temperatura (*topic* 2) y ha sido generada el 29 de febrero (*topic* 3). De esta forma, se ha establecido un nuevo modelo basado en *datastreams*: los clientes pueden estar interesados, y de hecho es lo más habitual, en una serie de medidas de una determinada índole en lugar de medidas esporádicas. Para satisfacer estas necesidades se ha desarrollado un nuevo tipo de operación, las suscripciones, que permiten a los compradores suscribirse a un determinado *topic* y recibir todas las medidas que pertenezcan a él.

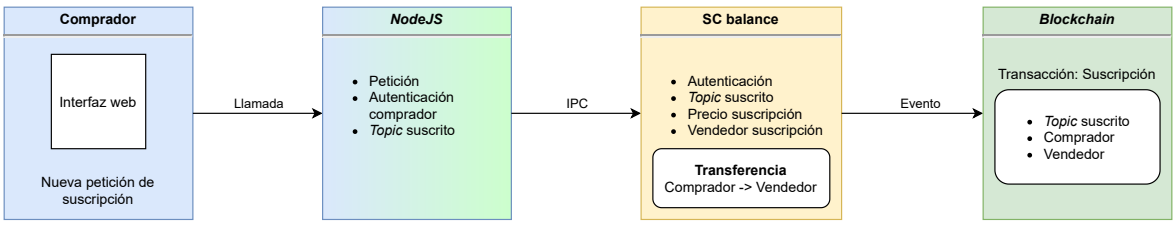


Figura 4.1: Esquema de flujo de una suscripción.

El esquema del flujo de ejecución de esta operación se muestra en la Figura 4.1. Se explican a continuación las fases de este flujo:

1. Un comprador, desde la interfaz web, observa las medidas registradas o los *topics* disponibles y decide que desea suscribirse a uno de ellos. Para esto, realiza una petición desde *JavaScript* especificando el *topic* en cuestión. Esta se recibe, a través de una llamada, en el núcleo del servidor web.

2. El *NodeJS*, que se comunica tanto con la interfaz web como con la *Blockchain* a través de un nodo de cliente, recibe la petición. Como el comprador tiene que estar autenticado para realizar la operación de suscripción, este componente puede extraer de manera sencilla los datos de autenticación. Además, lee en la petición el *topic* al que el cliente se desea suscribir. A través de la interfaz IPC, este componente realiza una llamada al SC balance con estos datos.

3. En este contrato se ejecuta la función *purchaseSubscription*, que genera una transferencia de *tokens* directa desde el comprador hacia el vendedor que ha registrado el *topic* en cuestión y le ha puesto precio a su suscripción. Esto genera un nuevo evento en la *Blockchain*.

4. Por último, se registra una transacción en la cadena del tipo suscripción, que contiene las direcciones del comprador y el vendedor, así como el *topic* al que se ha suscrito el primero.

Esta última transacción se puede utilizar como filtro para saber qué medidas se han generado antes y después de que un determinado comprador se suscribiera a un *topic*. Además, cabe destacar que las suscripciones solo necesitan una transacción y, a partir de ese momento, las medidas son adquiridas de manera implícita: no requieren transacciones adicionales. Como se discutirá en el Capítulo 5, esto ayuda considerablemente a la escalabilidad en tamaño de la *Blockchain*.

4.2 Cambios y mejoras en la implementación

En esta sección se va a realizar una segunda revisión a los SCs y componentes, centrándose, en este caso, en los cambios que se han realizado con el fin de soportar el nuevo modelo de suscripciones.

```
1     function addTopic(string memory topic, bytes32 key) public {}
2     function getTopics() public view returns (string[] memory) {}
3     function getIoTAddress(string topic) public view returns (address
          ) {}
```

Figura 4.2: Funciones más significativas del SC datos para el modelo de suscripciones.

4.2.1 *Smart Contracts*

De los tres contratos inteligentes que se han visto, dos de ellos han sido objeto de grandes cambios y actualizaciones para permitir las suscripciones a *datastreams*. El SC acceso no se ha modificado, ya que el registro y autorización de productores *IoT* al BIDM permanece tal como se describió en el Capítulo 3. Por el contrario, tanto el SC datos como el SC balance incluyen ahora nuevas funcionalidades que se describen a continuación.

SC datos

Se muestra en la Figura 4.2 un resumen con las primitivas de algunas de las funciones nuevas relacionadas con el nuevo modelo.

La función *addTopic* permite a un productor IoT registrar un nuevo *topic* en la *Blockchain*. Estos *topics* son descriptores de una característica básica de una medida. Por ejemplo, si el sensor X registra una medida de temperatura a las 13:30:00, se le podrían asignar los *topics* "temperatura" o "sensor X". La elección de *topics* se deja a discreción de los productores IoT para que se adapten lo mejor posible a su modelo de generación de datos. Por tanto, y volviendo a la explicación del SC, un productor puede registrar un *topic* nuevo mediante la función *addTopic*. Además de su nombre, también se registra una clave secreta de 32 bytes que servirá para cifrar las medidas pertenecientes a ese *topic*. Cuando una medida pertenezca a varios *topics*, se cifrará con una clave que está formada como la operación XOR bit a bit de todas las claves individuales de *topic*. El acceso a estas claves está restringido de tal forma que solo pueden acceder a ellas el administrador de la plataforma y el productor *IoT* que las generó. Por el contrario, los *topics* son públicos para facilitar a los consumidores la suscripción. Para ver los *topics* disponibles, se puede utilizar la función *getTopics*. Por último, con *getIoTAddress* se puede obtener la dirección *Ethereum* del productor que registró un determinado *topic*.

SC balance

El SC balance también ha sufrido actualizaciones significativas, y se presentan en la Figura 4.3 las primitivas de las funciones más relevantes del modelo de suscripciones a *datastreams*.

Las suscripciones son la principal fuente de interacción de los consumidores con los

```
1     function getPriceSubscription(string memory topic) public view
          returns(uint256) {}
2     function purchaseSubscription(string memory topic) public {}
3     function isSubscribed(address client, string memory topic) public
          view returns (bool) {}
```

Figura 4.3: Funciones más significativas del SC balance para el modelo de suscripciones.

topics que han registrado los productores y que asignan a sus medidas. Cada *topic* lleva asociado un precio de suscripción, que se puede leer llamando a la función *getPriceSubscription*, y que toma el modelo de negocio de tarifa plana. La inclusión de otros modelos de negocio, como suscripciones por tiempo o por número de medidas, sería fácilmente implementable pero no se ha realizado en este TFM ya que establecer un modelo de negocio viable no estaba entre sus objetivos. Para realizar la suscripción a un determinado *topic* los consumidores pueden hacer uso de la función *purchaseSubscription*, siempre y cuando tengan *tokens* suficientes. Una de las funciones informativas es *isSubscribed*, que devuelve un valor *true* si el usuario especificado está suscrito al *topic* especificado o *false* en caso contrario. Esta función tiene una gran utilidad para comprobar si un usuario tiene o no acceso a una medida.

4.2.2 Componentes

Además de los SCs, muchos de los componentes descritos en el Capítulo 3 también se han actualizado, incluyéndose en ellos nuevas funcionalidades relativas al modelo de suscripciones. En este apartado se describen estos cambios, centrando esta descripción en la relación que tienen los cambios implementados con las operaciones del BIDM, incluyendo las suscripciones.

Market core

Como se ha comentado al hablar de los *topics*, el encriptado de medidas al guardarlas en la base de datos ahora utiliza una clave formada como la XOR bit a bit de las claves de *topic*. Por tanto, el *Market core*, que es el componente encargado de la comunicación con la base de datos, es quien implementa estos cambios. Además, se ha desarrollado un servidor REST que cambia la forma de acceder a las medidas. Los usuarios pueden pedir las medidas completas a este servidor desde la interfaz web tras pasar por un proceso de autenticación. Con información sobre la dirección *Ethereum* de quien realiza la petición y el *hash* de la medida, este componente accede a la base de datos, desencripta la medida tras obtener sus *topics* en el SC datos, y se la envía al cliente a través de la interfaz web. Más información acerca del proceso de autenticación se detallará cuando se describan los cambios realizados a la interfaz web.

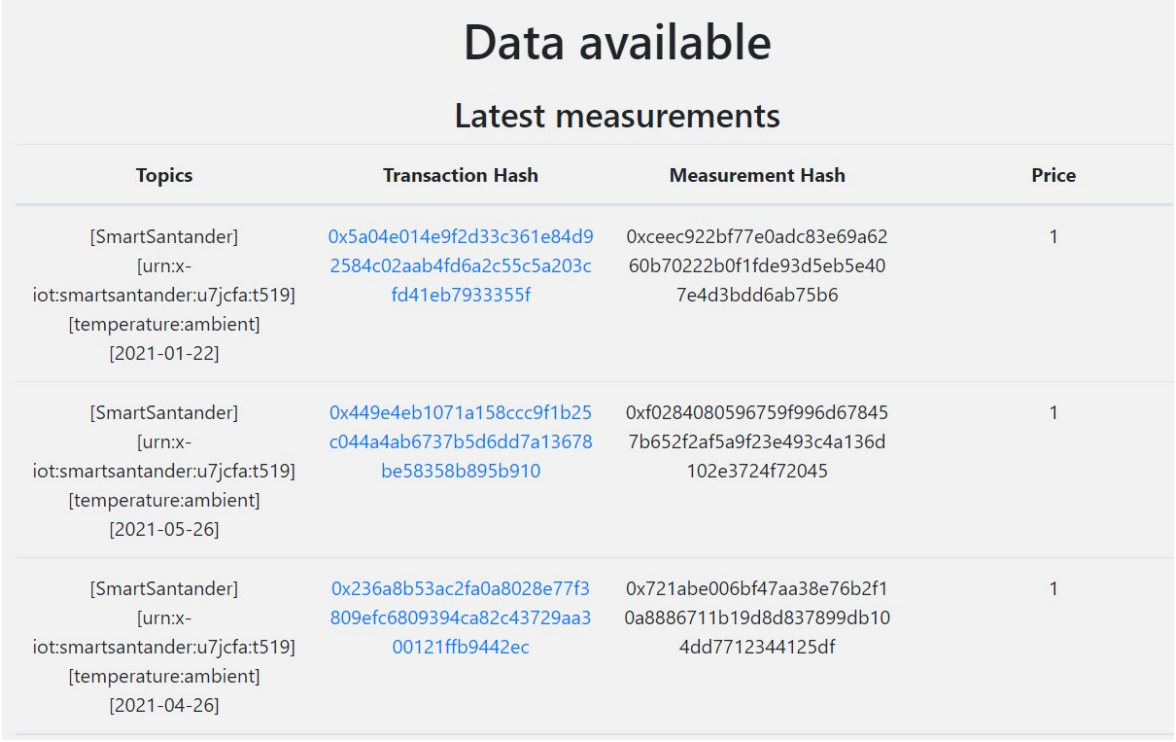


Figura 4.4: Ejemplo del resumen de tres medidas visualizadas en la interfaz web.

IoT proxy

En este componente, tanto la función *ProcessMeasurement* como *insertDataInBlockchain* han recibido cambios importantes centrados en la estructura de las medidas que se envían al SC datos para su registro. Se ha diseñado un sistema automático de registro de *topics*, de tal forma que todos los *topics* a los que pertenezca una medida que se va a guardar en la *Blockchain* se registran si no lo están ya. La clave de *topic* se genera a partir de un *Random Number Generator* (RNG) criptográficamente seguro, y el productor siempre puede leerla en el SC si la pierde en su almacenamiento local. Tras calcular la XOR bit a bit de todas las claves de *topic*, el componente guarda la medida cifrada en la clave de datos y posteriormente su hash, URL cifrada y lista de *topics* en la *Blockchain*.

Interfaz web

La interfaz web disponible para los compradores es el componente cuya funcionalidad más se ha extendido. Casi todas las rutas han visto modificado su aspecto para mostrar el listado de *topics* en lugar de descripción en las medidas, como se muestra en la Figura 4.4. Por otra parte, la ruta "Wallet" ya no es simplemente un buscador de las medidas compradas por el usuario, sino que incluye múltiples funcionalidades relacionadas con ambos modelos. En dos secciones separadas, el consumidor puede acceder

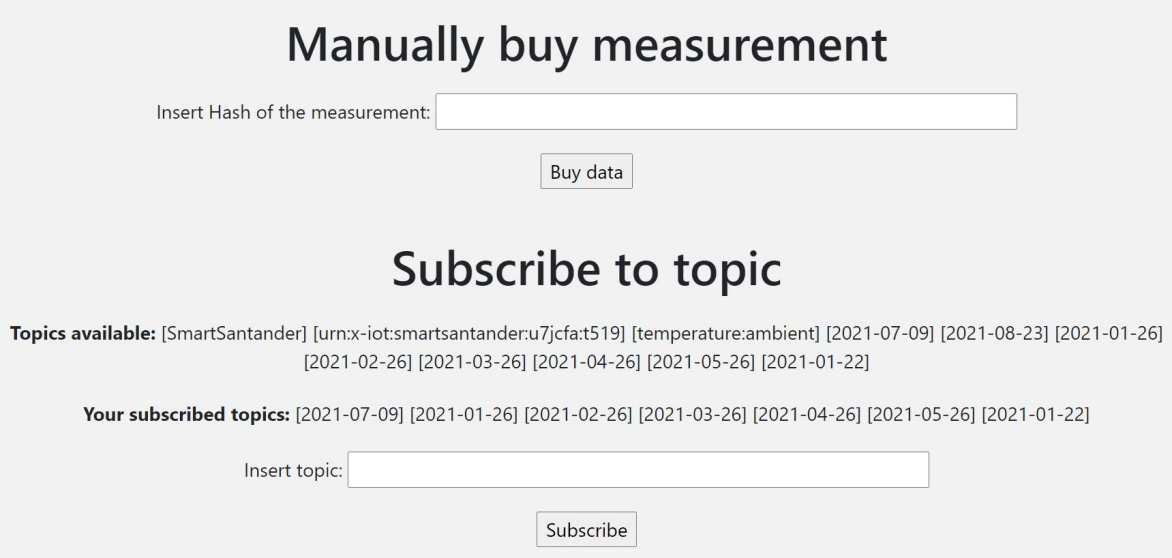


Figura 4.5: Sección del "Wallet" para realizar compras y suscripciones en la interfaz web. Se muestran *topics* de ejemplo.

a las medidas compradas individualmente y a las medidas pertenecientes a un *topic* al que está suscrito. Por otra parte, en esta ruta los consumidores pueden realizar las operaciones de compra y de suscripción, y se muestran en todo momento la lista completa de *topics* y los *topics* a los que el usuario ya está suscrito para mayor comodidad. Esta sección se muestra en la Figura 4.5.

En cuanto a la visualización de medidas, se ha incluido una funcionalidad que aprovecha el inicio de sesión para realizar la autenticación de los usuarios. Al utilizar la dirección *Ethereum* como nombre de cuenta, y la contraseña es la misma que se utiliza para desbloquear dicha cuenta en la *Blockchain*, iniciar sesión en el servidor web implica estar autenticado en la *Blockchain*. Esta característica se ha utilizado para comprobar la autorización de los usuarios para acceder al valor de las medidas. Solo aquellos clientes que tienen el acceso permitido pueden realizar peticiones desde la web hasta el *Market core*, que es quien accede a la base de datos y devuelve las medidas, como se ha comentado previamente. El servidor web comprueba si una cuenta tiene acceso a la medida seleccionada a través de las funciones *isSubscribed* del SC balance y de los eventos pasados generados en la *Blockchain*, tanto de compra como de suscripción.

Capítulo 5

Análisis y resultados

En este capítulo se exponen los resultados de los análisis realizados sobre la plataforma. Se han colocado sellos de tiempo en varios puntos clave del código de los componentes con el fin de observar el retardo que sufren las operaciones clave desde la petición hasta el final del proceso de ejecución. Además, se han tomado medidas del tamaño en disco de la *Blockchain* y de la base de datos para ver cómo evolucionan según se van registrando medidas y compras. El análisis de comportamiento que se ha llevado a cabo se ha parametrizado en función de diferentes variables, lo que ha permitido comprobar el rendimiento del BIDM en diferentes condiciones de carga. Además, se han realizado pruebas tanto con medidas generadas sintéticamente bajo diferentes distribuciones como con medidas reales generadas por *SmartSantander*. Finalmente, se van a presentar los mismos resultados con el modelo de suscripciones y se va a realizar una comparativa entre ambas filosofías.

5.1 Integración con *SmartSantander*

Hasta este momento, se ha hablado de comunicar productores y consumidores de cualquier naturaleza a través de los sistemas descritos. En la prueba de concepto, se utilizaron medidas de ensayo creadas manualmente y enviadas mediante consola de comandos, de tal forma que la funcionalidad quedaba comprobada. Sin embargo, también es preciso demostrar que el BIDM funciona bajo condiciones reales. Para ello, se ha desarrollado un sistema que conecta el BIDM con *SmartSantander* y registra a la plataforma IoT que gestiona el despliegue IoT en Santander como uno de los productores IoT del BIDM. Por tanto, las medidas generadas por esta plataforma pasan por los procesos de registro descritos en el Capítulo 3 y los consumidores pueden comprarlas o suscribirse a ellas. A continuación, se describen los nuevos componentes y modificaciones que se han realizado para poder llevar a la práctica esta integración.

- Para implementar la integración de *SmartSantander* al BIDM se ha hecho uso del API de suscripciones de dicha plataforma [31]. Entre las opciones soportadas por este API está la de especificar un servidor de *callback* que reciba las notificaciones

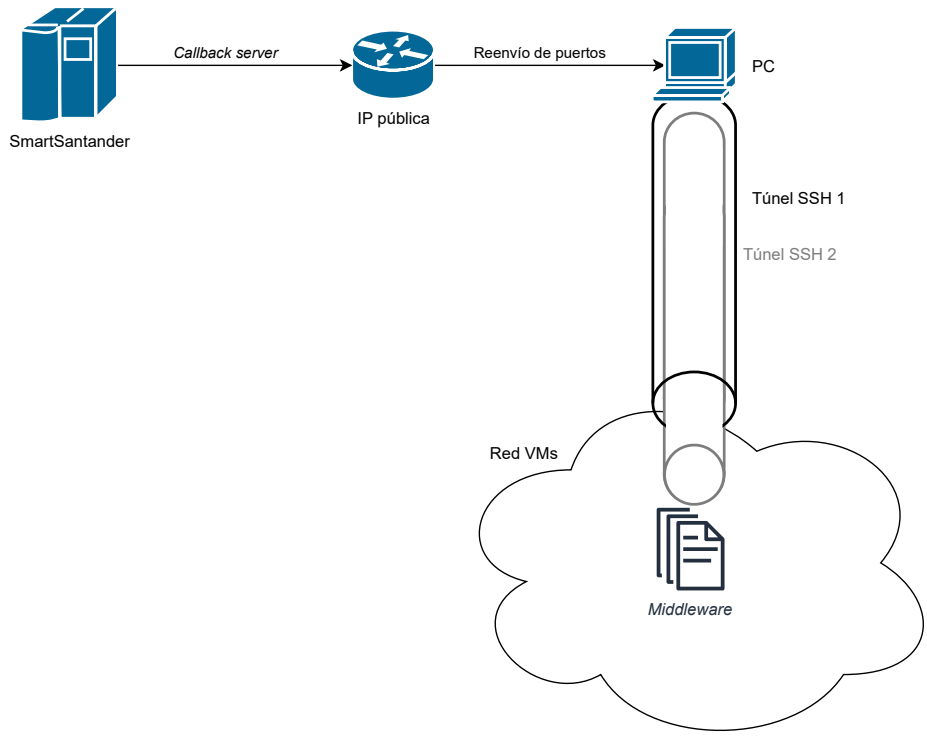


Figura 5.1: Conexión *SmartSantander* - *Middleware*.

de las medidas que se generan por la infraestructura IoT de *SmartSantander*. Este servidor es el *Middleware* que se ha descrito en capítulos anteriores como parte de la plataforma desarrollada.

- El *Middleware* tiene como función recibir las medidas de *SmartSantander*. En el despliegue realizado en este TFM, dicho componente se encontraba en una red privada. Para que el acceso fuera posible, se concatenaron múltiples túneles SSH, tal como se muestra en la Figura 5.1. La plataforma de *SmartSantander* envía las medidas a una interfaz pública, que mediante reenvío de puertos se redirigen a un PC. Este PC está conectado mediante el túnel SSH 1 a la red de máquinas virtuales en la que se realiza todo el procesado. El *Middleware*, no obstante, está esperando en un puerto específico, lo que requiere un nuevo túnel SSH dentro del anterior. De esta forma, las medidas generadas por el productor *IoT* son redirigidas múltiples veces hasta que pueden llegar a su destino.

- Una vez la medida está en el *Middleware*, se transforma al modelo de datos utilizado en el BIDM y se realiza su registro en el mismo a través del módulo *Market*. Este proceso se ha implementado a través de un *script Python*. Tras enviar la medida al *IoT proxy*, ya no existe diferencia con el resto en términos de funcionalidad, provengan de otro productor o de ensayos con medidas generadas manualmente.

Esta integración hace posible el análisis de rendimiento del BIDM bajo condiciones reales, ya que las medidas provendrán de una infraestructura real como *SmartSantander*. Además, para poder llevar a cabo un análisis del rendimiento del BIDM bajo diferentes tipos de condiciones, también se ha desarrollado un generador de medidas sintéticas con el que es posible parametrizar tanto el tamaño de éstas como la forma en la que se generan (fundamentalmente la distribución entre los tiempos de generación de las medidas consecutivas).

5.2 Metodología de análisis

Esta sección describe el proceso que se ha seguido para llevar a cabo el análisis de rendimiento del BIDM. En los Capítulos 3 y 4 se han detallado los flujos que sigue la información en las operaciones más relevantes, como el registro o la compra de medidas. Para poder analizar el retardo de cada operación, es necesario colocar "sondas" que registren el sello de tiempo en los distintos componentes involucrados en estos procesos y que se están ejecutando concurrentemente. Cada una de estas sondas accede a un fichero de registro en el que se escriben las marcas de tiempo (*timestamps*) en las que suceden los distintos pasos de estos procesos. En el proceso de registro, los pasos cuya marca de tiempo queda registrada son:

- El productor inicia la operación enviando la nueva medida al *IoT proxy*.
- Este componente comprueba que el productor tiene acceso y marca otro *timestamp*.
- Tras el procesado de la medida, y justo antes de enviarla a la base de datos.
- Tras recibir confirmación del registro correcto en la base de datos.
- Se extrae la información de la medida que se va a registrar en la *Blockchain*, se comprueba su precio y se envía al SC datos para realizar este registro.
- Un último *timestamp* tras finalizar este proceso de registro de la medida en el BIDM, incluyendo su precio.

Por otra parte, en el proceso de compra se han establecido los siguientes hitos cuya marca de tiempo queda registrada:

- El consumidor inicia el proceso lanzando la petición de compra de una medida.
- Tras procesar dicha petición, el servidor web la envía a la *Blockchain*.
- Escuchando el evento generado en la *Blockchain*, el *Market* detecta la compra. Este *timestamp* se genera en ambos componentes, para asegurar el sincronismo obteniendo la diferencia entre ambos.

Tabla 5.1: Número de muestras.

Intervalos regulares	*Poisson*	*Smart Santander*	Longitud	Escalabilidad
5000 x4	5000 x4	>10000	5000 x3	>500000

- El *Market* realiza su procesado, y registra un último sello de tiempo tras finalizar la transacción de confirmación en la *Blockchain*.

Los ficheros generados consisten en una secuencia de sellos de tiempo, identificadores de cada medida y de la sonda que lo ha escrito. De esta forma, se pueden procesar automáticamente para obtener el retardo de cada una de las fases de cada operación. El análisis de estos ficheros se ha realizado mediante un *script Python*, que lee línea a línea y escribe los retardos obtenidos en otro fichero en forma de matriz. Este formato es legible directamente con *Matlab*, que es el programa utilizado para completar el análisis y generar las gráficas correspondientes.

En las próximas secciones se presentan los resultados obtenidos con los distintos tipos de generación de medidas y parametrización de variables. En primer lugar se muestran los resultados del análisis de los procesos de registro y compra de manera individual, y a continuación, aquellos relacionados con el nuevo modelo de *datastreams*. Se finalizará con una comparativa y una discusión al respecto.

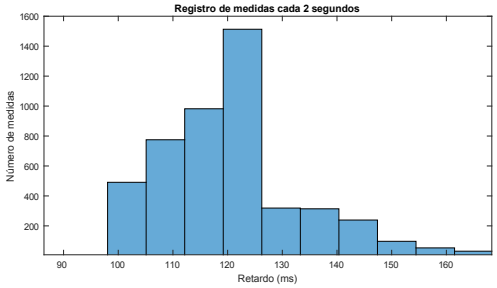
5.3 Compra y registro de medidas

El primero de los análisis está realizado sobre las operaciones de registro y compra de medidas. En cada una de las subsecciones se presenta y discute cada una de las diferentes pruebas realizadas y los resultados obtenidos. En la Tabla 5.1 se resumen el número de muestras tomadas en cada uno de los experimentos.

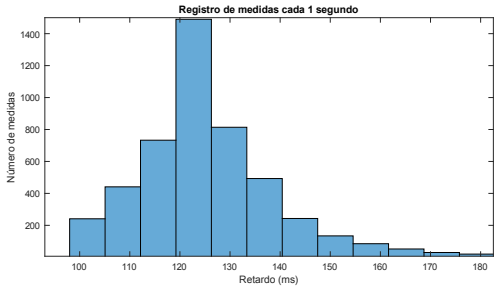
5.3.1 A intervalos regulares

Para este escenario de análisis, se ha utilizado el generador sintético para generar y registrar medidas a intervalos regulares. Para el proceso de compra, se ha hecho uso de la función automatizada del *Market* que realiza compras regularmente desde una determinada cuenta de cliente. El objetivo de este experimento es observar la influencia del parámetro tiempo entre medidas sobre el retardo en el rendimiento del BIDM. Para ello, y como se observa en la Tabla 5.1, se han tomado 5000 muestras en 4 casos: separando los registros y las compras de medidas 2 segundos, 1 segundo, 0.5 segundos y 0.2 segundos, respectivamente. Los resultados de retardo obtenidos se pueden ver en la Figura 5.2 para el caso del registro de nuevas medidas, y en la Figura 5.3 para la compra. En todos los casos se ha trabajado con medidas con un tamaño de 229 bytes.

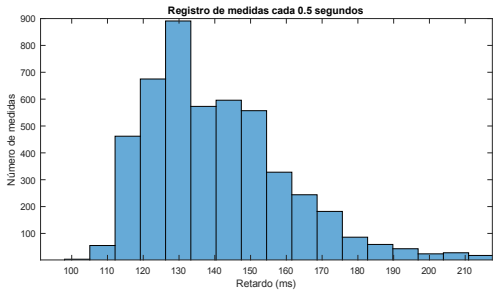
En ambos casos, se muestran los histogramas de retardo de cada operación en cada uno de los 4 casos mencionados. En el caso del registro de medidas, las operaciones



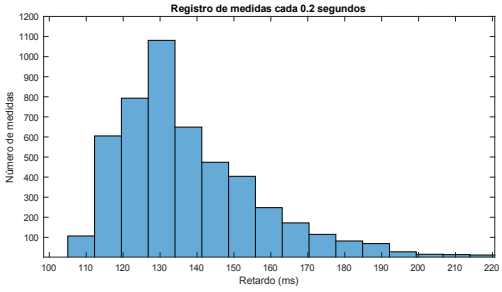
(a) 2 segundos entre medidas



(b) 1 segundo entre medidas

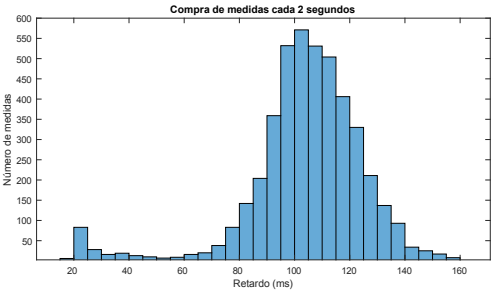


(c) 0.5 segundos entre medidas

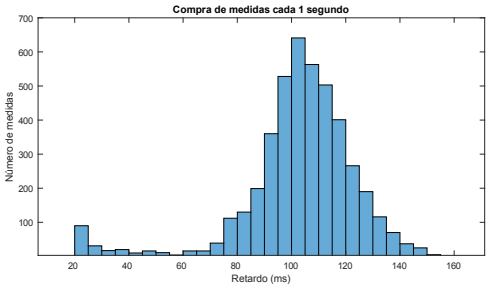


(d) 0.2 segundos entre medidas

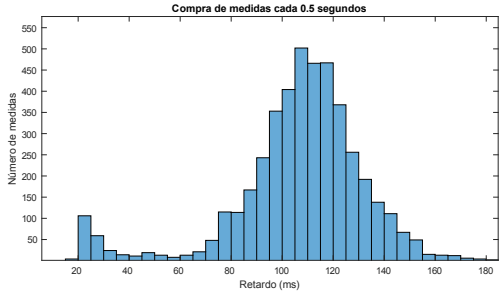
Figura 5.2: Resultados de retardo para el registro de medidas a intervalos regulares.



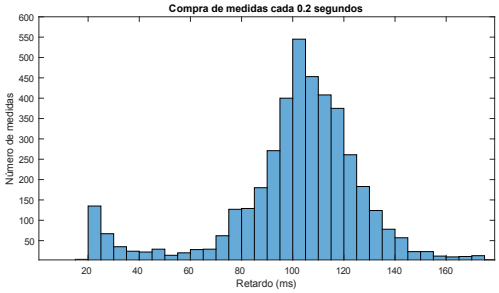
(a) 2 segundos entre medidas



(b) 1 segundo entre medidas



(c) 0.5 segundos entre medidas



(d) 0.2 segundos entre medidas

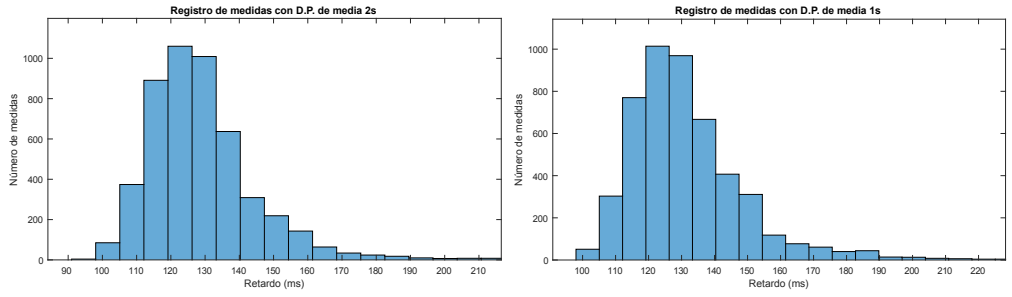
Figura 5.3: Resultados de retardo para la compra de medidas a intervalos regulares.

tardan una media de 120-130 milisegundos en completarse, con una varianza de $5{,}0 \cdot 10^4$ ms^2; mientras que en la compra de medidas, la media es ligeramente superior a los 100 milisegundos y la varianza es de $6{,}5 \cdot 10^2$ ms^2. Se observa que, como en gran parte de los procesos naturales, la distribución estadística recuerda a una normal o gaussiana, con una media claramente identificable y "colas" a ambos lados. En estos casos, la cola por la parte inferior es más pronunciada ya que resulta imposible que el procesado se realice en un tiempo muy inferior a la media. Otro detalle importante es la diferencia entre los 4 casos de una misma operación. Podría parecer lógico pensar que si las operaciones llegan con una tasa superior, el procesado se empezaría a estresar y ralentizarse. Sin embargo, se aprecia que las diferencias entre unos casos y otros son prácticamente inapreciables o incluso inexistentes. Esto permite concluir que el BIDM es capaz de soportar operaciones sobre la *Blockchain* a una alta tasa sin dar problemas de rendimiento.

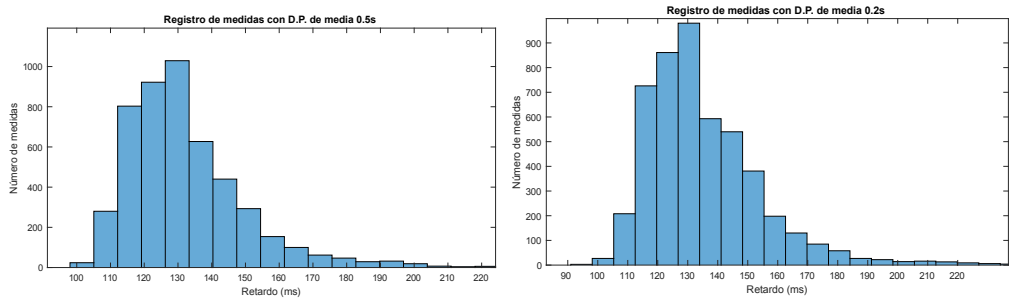
5.3.2 Con distribución de *Poisson*

En este escenario de análisis, una vez más, se generan medidas utilizando el generador sintético. Sin embargo, en este caso el tiempo entre generaciones ya no es constante, sino que sigue una distribución de *Poisson*. Lo mismo se ha configurado para el proceso de compra automatizado. El objetivo de este experimento es similar al anterior, pero con una diferencia importante; como el tiempo entre nuevos eventos es aleatorio, ahora va a haber operaciones que lleguen cuando aún no se ha terminado de procesar la anterior, y por tanto va a existir concurrencia. Como se ha visto en la Tabla 5.1, se han tomado 5000 muestras en 4 casos utilizando todos ellos una distribución de *Poisson* para el registro y la compra de medidas donde la media entre llegadas de nuevos eventos es de 2 segundos, 1 segundo, 0.5 segundos y 0.2 segundos respectivamente. Los resultados de retardo obtenidos se pueden observar en la Figura 5.4 para el caso del registro de nuevas medidas, y en la Figura 5.5 para la compra.

Al igual que en el escenario anterior, se muestran los histogramas de retardo para los 4 casos. De nuevo, la media en el caso de los registros está en torno a los 120-130 ms (con una varianza de $4{,}3 \cdot 10^4$) que va creciendo muy ligeramente según disminuye la media de la distribución de tiempo entre dos medidas consecutivas. En las operaciones de compra, la media se mantiene ligeramente superior a los 100 ms, con una varianza de $1{,}4 \cdot 10^3$ ms^2. Además de lo que ya se ha comentado en el caso de los eventos a intervalos regulares, hay una diferencia principal apreciable en el caso de las compras de medidas que puede resultar contraria a lo intuitivo. Se observa que la aparición de medidas cuyo procesado ocurre en un tiempo muy corto es superior en los casos donde las compras llegan con una tasa mayor. La explicación de este fenómeno se encuentra en el funcionamiento de la *Blockchain* y el mecanismo de *Proof of Authority* o PoA. En general, cuando una transacción ocurre en la *Blockchain*, los nodos mineros trabajan en ella hasta conseguir minar un bloque. Cuando esto ocurre, todas las transacciones pendientes se incluyen en ese bloque, que se añade a la *Blockchain*. En el mecanismo PoA utilizado en esta plataforma, el minado es prácticamente instantáneo ya que no

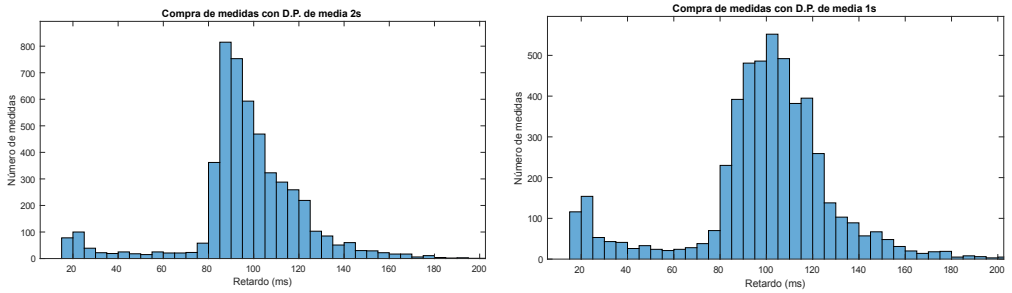


(a) Media 2 segundos entre medidas (b) Media 1 segundo entre medidas

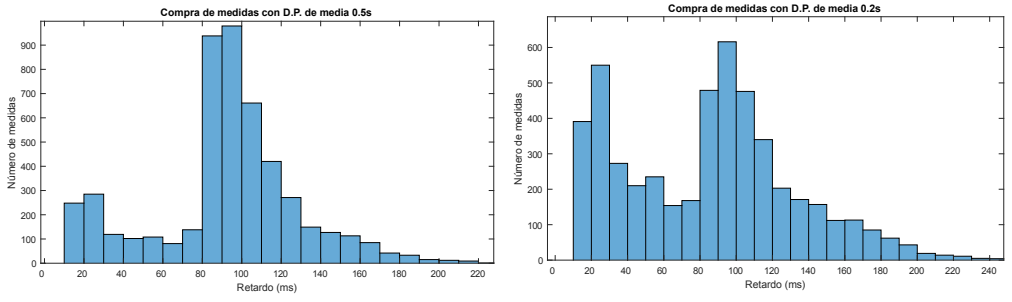


(c) Media 0.5 segundos entre medidas (d) Media 0.2 segundos entre medidas

Figura 5.4: Resultados de retardo en el registro de medidas con distribución de *Poisson*.



(a) Media 2 segundos entre medidas (b) Media 1 segundo entre medidas



(c) Media 0.5 segundos entre medidas (d) Media 0.2 segundos entre medidas

Figura 5.5: Resultados de retardo en la compra de medidas con distribución de *Poisson*.

existe el incentivo de obtener beneficio económico a cambio de minar, y por tanto la dificultad del minado ha sido reducida al mínimo. En todos los casos anteriores, cada vez que ocurre una transacción, parte del retardo se debe a la validación y minado del bloque, aunque es un efecto prácticamente inapreciable. Sin embargo, cuando se utiliza una distribución de *Poisson* y comienzan a solaparse los procesos de compra, los bloques empiezan a incluir varias transacciones de manera automática. Esto provoca un efecto secundario beneficioso, ya que hay transacciones que no necesitan esperar a la validación y el minado de un nuevo bloque, porque ya tienen uno esperando. Esto es lo que se observa en las gráficas de compra, donde hay más medidas con un procesado corto cuanto más rápida es la tasa de llegada. Por lo tanto, se puede concluir que el sistema tiene un comportamiento, si cabe, mejor cuanto más demandado se encuentre.

5.3.3 *Smart Santander*

Para este escenario, no se utilizará el generador de medidas sintéticas sino que se va a hacer uso de la integración con *SmartSantander*, siendo este el productor de medidas IoT utilizado para el análisis de rendimiento. Debido a la naturaleza de la plataforma IoT, cabe esperar que presente un comportamiento similar a las medidas generadas con distribución de *Poisson*, ya que la principal característica de esta es que modela sistemas de este tipo. Con este experimento quedará demostrado el funcionamiento de la *Blockchain* bajo condiciones reales de producción de medidas por una plataforma IoT, así como sus valores de retardo. Se han generado alrededor de 10000 operaciones de registro y compra, aunque esta segunda no presenta ningún cambio respecto a los experimentos anteriores. Los resultados obtenidos se muestran en la Figura 5.6 para el caso del registro de nuevas medidas generadas por *SmartSantander*, y en la Figura 5.7 para la compra.

Se puede apreciar que tanto el retardo medio como la distribución del mismo en forma de histograma son muy similares a los vistos en los casos anteriores. Esto demuestra que el rendimiento del BIDM no presenta cambios aparentes si se utiliza bajo condiciones realistas de generación de medidas. El contenido de las medidas como tal no es diferente al del generador automático, ya que para su programación se utilizó una medida de *Smart Santander* como modelo. El hecho de que no haya mucho más que comentar es un resultado en sí mismo: la plataforma está preparada para ser integrada con productores IoT reales.

5.3.4 Longitud variable

Como se ha descrito en todos los escenarios anteriores se ha ido variando el parámetro tiempo entre dos medidas consecutivas, pero el tamaño de las medidas utilizadas era constante. En este caso, el objetivo es determinar si una variación en las medidas registradas y compradas afecta al retardo a la hora de procesarlas. Se han generado tres grupos de 5000 medidas de forma sintética, separando las operaciones en el tiempo en intervalos fijos de 1 segundo. Los tres grupos de medidas se dividen según su longitud.

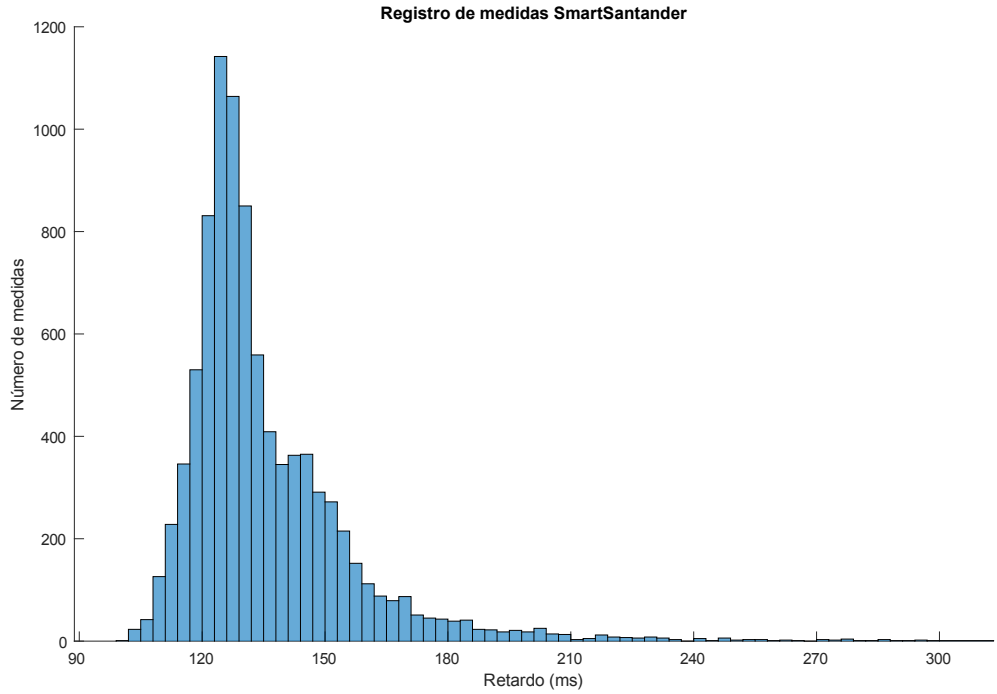


Figura 5.6: Resultados de retardo para el registro de medidas generadas por *Smart Santander*.

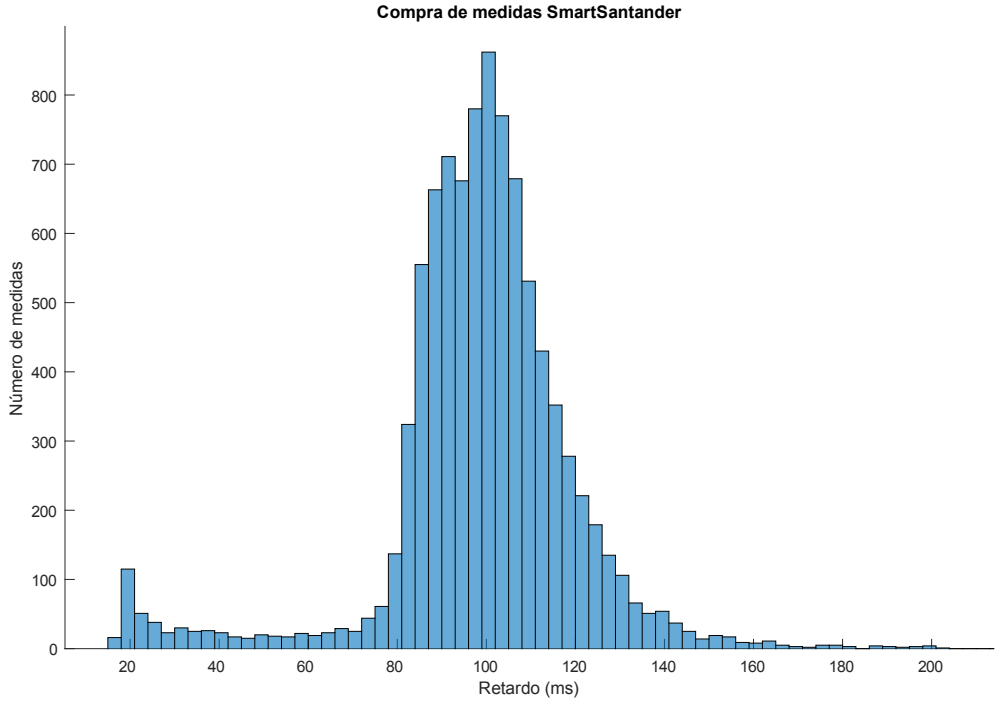
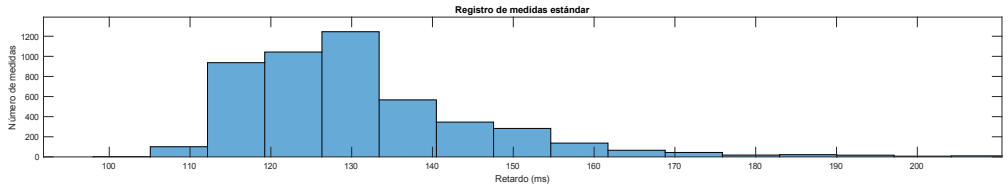
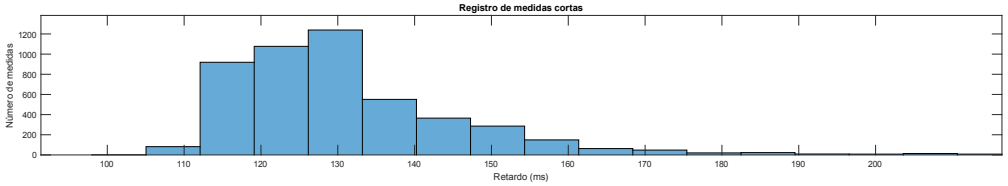


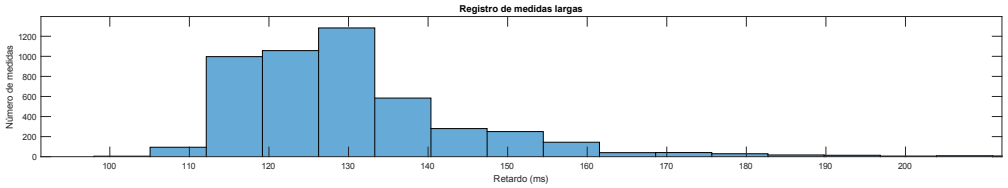
Figura 5.7: Resultados de retardo para la compra de medidas generadas por *Smart Santander*.



(a) Medidas estándar - 229 bytes



(b) Medidas cortas - 55 bytes



(c) Medidas largas - 997 bytes

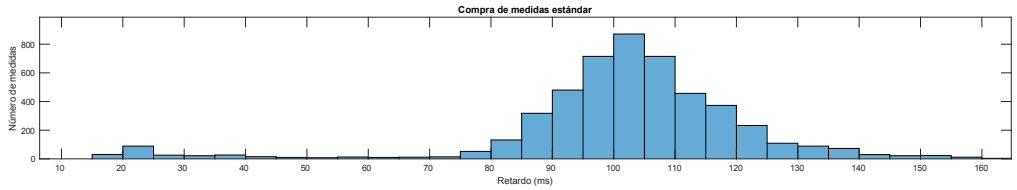
Figura 5.8: Resultados de retardo para el registro de medidas con diferentes longitudes.

Las **medidas cortas** tienen un contenido de 55 bytes, las **medidas estándar** ocupan 229 bytes, y el cuerpo de las **medidas largas** es de 997 bytes. En un principio, el procesado no depende de la longitud de las medidas, pero es necesario considerar la posibilidad de que existan variaciones debido al funcionamiento interno de la *Blockchain*. Comprobar si esta posibilidad es real es la motivación de este experimento. Los resultados de retardo obtenidos se muestran en la Figura 5.8 para el caso del registro de nuevas medidas, y en la Figura 5.9 para la compra.

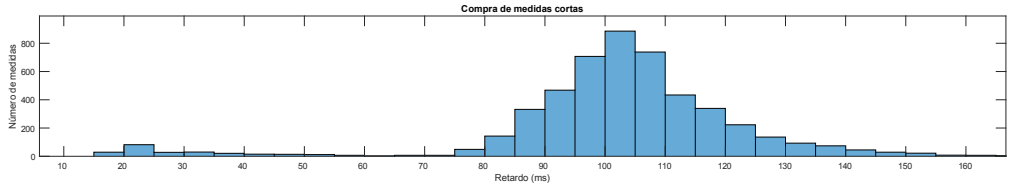
En esta ocasión, los histogramas revelan que la influencia del tamaño de las medidas es nula. Exceptuando ínfimas variaciones debido a la naturaleza estadística del análisis, tanto los registros como las compras de medidas muestran un comportamiento prácticamente idéntico a pesar de que un grupo de medidas tenga más de 10 veces la longitud de otro.

5.3.5 Escalabilidad en tiempo

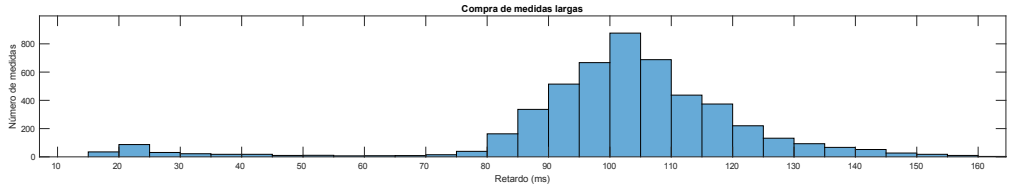
Una de las características más importantes de un sistema como el que se ha desarrollado en este TFM es su capacidad para mantener su rendimiento a medida que aumenta su carga, la escalabilidad. El siguiente análisis experimental consiste en registrar y comprar cantidades masivas de medidas, y comprobar si tras varios días, con una *Blockchain* que contiene varios cientos de miles de bloques y una base de datos con aproximadamente el



(a) Medidas estándar - 229 bytes



(b) Medidas cortas - 55 bytes



(c) Medidas largas - 997 bytes

Figura 5.9: Resultados de retardo para la compra de medidas con diferentes longitudes.

mismo número de medidas, el BIDM se comporta en términos de retardo como lo hacía al inicio. Antes de realizar estas pruebas se ha reiniciado la *Blockchain* completamente, para asegurar que los resultados correspondientes a las primeras operaciones tienen lugar con una cadena y base de datos tan vacías como sea posible. Para realizar los registros se ha utilizado el generador automático con 0.5 segundos de intervalo fijo entre operaciones, y las compras siguen un proceso análogo con su correspondiente sistema automático. Como se describe en la Tabla 5.1, para este experimento se han realizado más de 500000 operaciones, fruto de varios días dejando trabajar al BIDM de forma automática. Los resultados de retardo obtenidos se pueden observar en la Figura 5.10 para el registro de medidas, y en la Figura 5.11 para la compra.

Obviando las ligeras variaciones debidas a la aleatoriedad del experimento, el resultado del mismo es fácilmente apreciable. El eje de ordenadas se corresponde con el retardo de cada operación, que se mantiene constante con la evolución del eje de abscisas. Este último representa el número de bloques en la cadena, que es equivalente al número de operaciones previamente realizadas y que podrían, potencialmente, deteriorar el rendimiento del BIDM. Como se puede observar, este deterioro es inexistente y las operaciones presentan un retardo constante, independientemente de la carga previa en el sistema. Cabe mencionar que no es posible asegurar que no exista un punto en el futuro, cuando la plataforma esté cargada con un número muy superior de medidas, en el que el rendimiento comience a deteriorarse. Sin embargo, se puede asumir que,

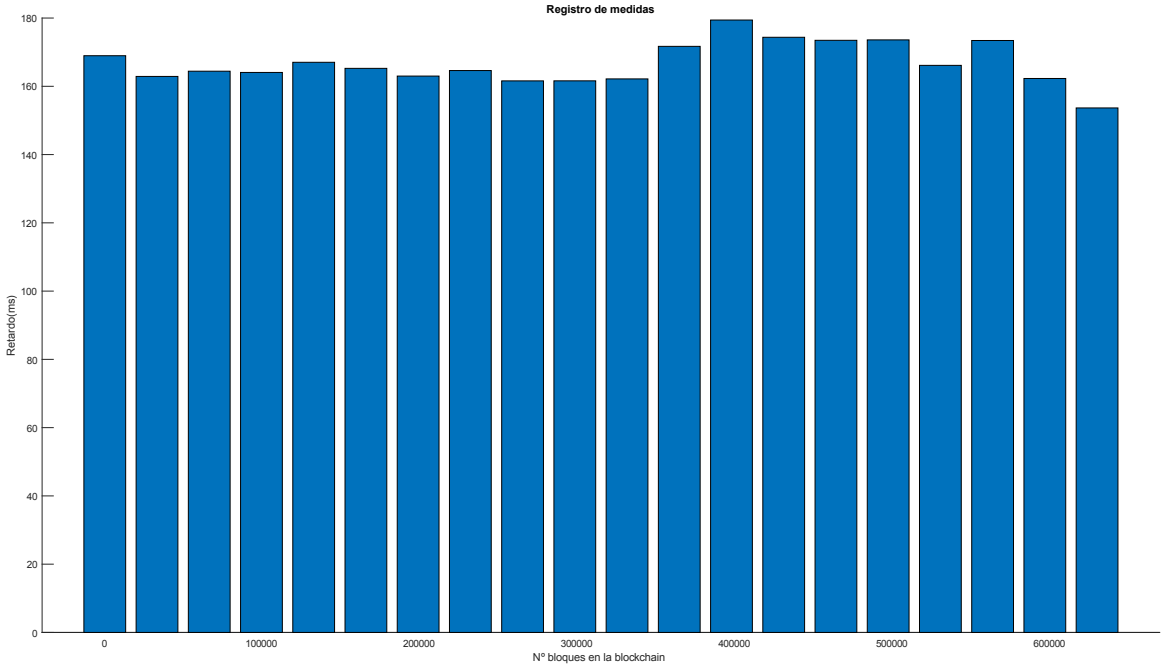


Figura 5.10: Resultados de retardo para el registro de medidas según la carga de la *Blockchain*.

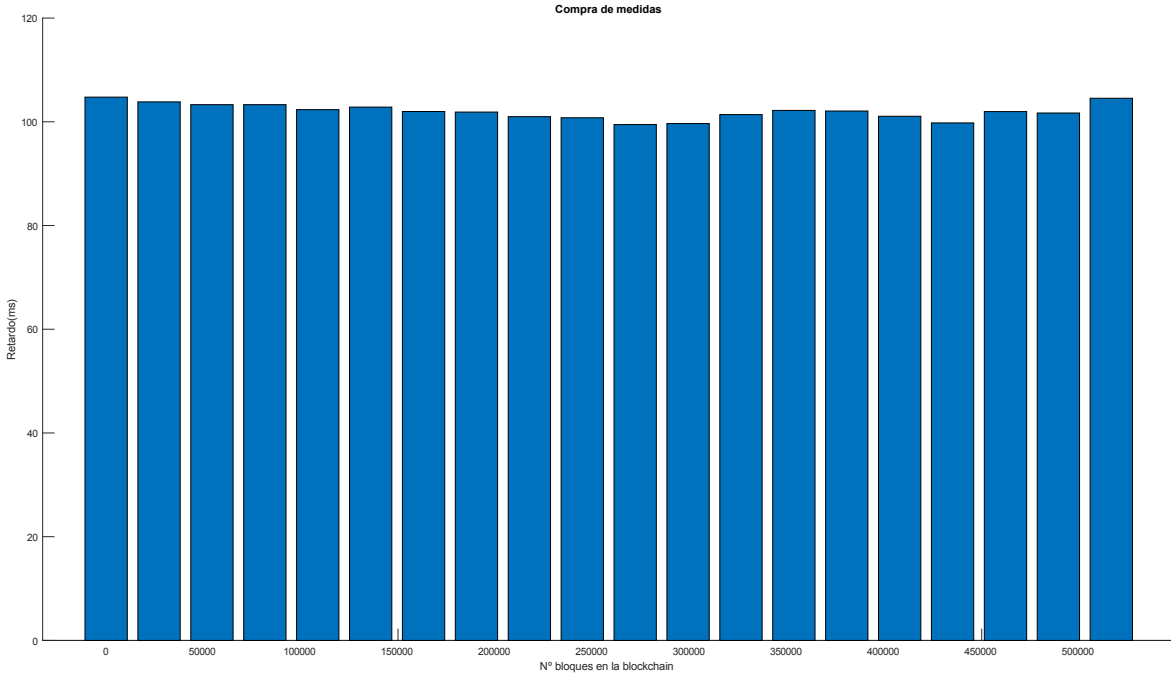


Figura 5.11: Resultados de retardo para la compra de medidas según la carga de la *Blockchain*.

Tabla 5.2: Retardo por secciones para el registro de medidas.

Autenticación	Procesado I	Registro IPFS	Procesado II	Registro *Blockchain*
1.29 %	0.29 %	87.55 %	6.31 %	4.57 %

Tabla 5.3: Retardo por secciones para la compra de medidas.

Procesado	Compra *Blockchain*	Confirmación *Blockchain*
1.19 %	16.32 %	82.49 %

a la vista del experimento realizado, este punto ficticio no existe o no es razonable alcanzarlo haciendo un uso natural del BIDM.

5.3.6 Retardo por secciones

En este apartado, se ha hecho una separación del retardo de cada operación aprovechando las diferentes sondas colocadas dentro de los componentes. El objetivo es analizar el efecto de cada una de las fases de la operación, para ver cuales generan más retardo. Las muestras que se han utilizado son las que se tomaron en los apartados de intervalos regulares y distribución de *Poisson*. Como ya se explicó en la Sección 5.2, el proceso de registro tiene seis sondas colocadas, lo que resulta en cinco divisiones, y el proceso de compra está dividido en tres secciones. Se muestran los resultados de este experimento en la Tabla 5.2 para el registro de medidas, y en la Tabla 5.3 para la compra.

Los porcentajes mostrados están calculados sobre el retardo total analizado en los apartados anteriores. Se ha considerado que el análisis de 40000 operaciones de cada tipo es suficiente para obtener unos resultados estadísticamente precisos. Como se puede observar, en el caso del registro la gran mayoría del retardo se debe a la base de datos IPFS, y es un tiempo que es prácticamente imposible de eliminar. En el proceso de compra, el retardo está repartido entre las fases que tienen interacción con la *Blockchain*. Cabe recordar que en la fase denominada "confirmación *Blockchain*" están incluidas dos transacciones, además de un acceso a la base de datos. Por tanto, una conclusión que se puede extraer de este análisis es que el retardo del sistema se debe, en su mayoría, a accesos necesarios a la base de datos, especialmente en la operación de registro de medidas.

5.4 Modelo de *datastreams*

Esta sección se centra en el análisis del nuevo modelo de suscripciones, su rendimiento y una comparativa con el modelo tradicional de compras de medidas individuales. En el Capítulo 4 se vio la operación de suscripción y lo que implica. Un comprador puede, en lugar de adquirir cada medida por separado, suscribirse a un flujo de medidas y

obtener de manera implícita todas las que formen parte de este. A nivel de retardo, este modelo presenta una ventaja conceptual. Aunque la suscripción pueda ser una operación más costosa, solo se tiene que realizar una vez. Por tanto, el retardo de una suscripción es un valor con mucho menos impacto sobre el rendimiento final de la plataforma, ya que solo va a ser ejecutada en contadas ocasiones. Sin embargo, se han realizado análisis sobre esta operación de todas formas. En esta sección se van a presentar dichos análisis, se va a discutir el impacto de todas las operaciones sobre el espacio en disco, y finalmente, se va a realizar una comparativa entre el modelo de suscripciones y el de compras individuales.

5.4.1 Resultados de retardo

En primer lugar se analiza el retardo de la operación de suscripción, aunque ya se ha comentado que su impacto es inferior al de otras operaciones. Se han tomado 100 *timestamps* al inicio y al final del proceso, con el fin de obtener un resultado razonablemente fiable. Si bien su comportamiento estadístico no es tan interesante como el de las operaciones que se repiten constantemente, se observa que el retardo de una suscripción se encuentra en el rango de entre 100 y 200 milisegundos, con una media de aproximadamente **140 ms** (y una varianza de $2{,}0 \cdot 10^4$ ms^2). Por tanto, a pesar de ser una operación que se realiza escasas veces, su coste computacional y consiguiente retardo es similar al de las operaciones de registro y compra, si bien solo ligeramente superior.

Otro resultado interesante que no se ha contemplado hasta ahora es la experiencia del usuario al navegar por la interfaz web. Buena parte de esta depende de la eficiencia del navegador y las especificaciones del dispositivo con el que se accede, y es por esto que no se le ha prestado especial atención a lo largo de este capítulo. Sin embargo, ahora que existen dos modelos diferentes a comparar, puede resultar interesante observar cual de ellos se comporta mejor. Para ello, se ha medido el retardo que experimenta un usuario estándar utilizando un navegador comercial altamente extendido y un procesador Intel® Core™i5-6600K. Las muestras que se han utilizado para este experimento son a la hora de mostrar en pantalla las medidas compradas por el usuario. En el caso del modelo de compra estándar de medidas, el servidor web llama al SC balance y filtra los eventos de compra confirmada desde la dirección *Ethereum* del usuario que ha iniciado sesión. Esto implica recorrer toda la lista de eventos del SC. En el modelo de suscripciones, el servidor llama al SC balance y filtra los eventos de suscripción desde la dirección del usuario, que son menos numerosos. Una vez el programa sabe en qué momento el usuario se suscribió a un determinado *topic*, llama al SC datos y filtra los eventos de registro donde al menos uno de los *topics* asociados a la medida coincidan con aquellos a los que el usuario está suscrito. A primera vista puede parecer que el segundo modelo conlleva una mayor carga computacional, pero las listas que se recorren son más pequeñas y los filtros más eficientes. Una vez más, se han tomado 100 muestras del tiempo de carga de ambas visualizaciones: medidas compradas con el método tradicional y medidas compradas implícitamente a través de una suscripción.

Se puede observar que el retardo en el primer caso tiene una media de **1310 ms**, mientras que en el modelo de suscripciones es de **920 ms**. Se puede apreciar que el segundo proceso es casi un 50 % más rápido, si bien ambos son tiempos de carga relativamente normales en comparación con páginas web comerciales estándar. En cualquier caso, el resultado más destacable es que el modelo de suscripciones presenta un retardo en la búsqueda y filtrado de medidas inferior al modelo de compras tradicional.

5.4.2 Escalabilidad en espacio

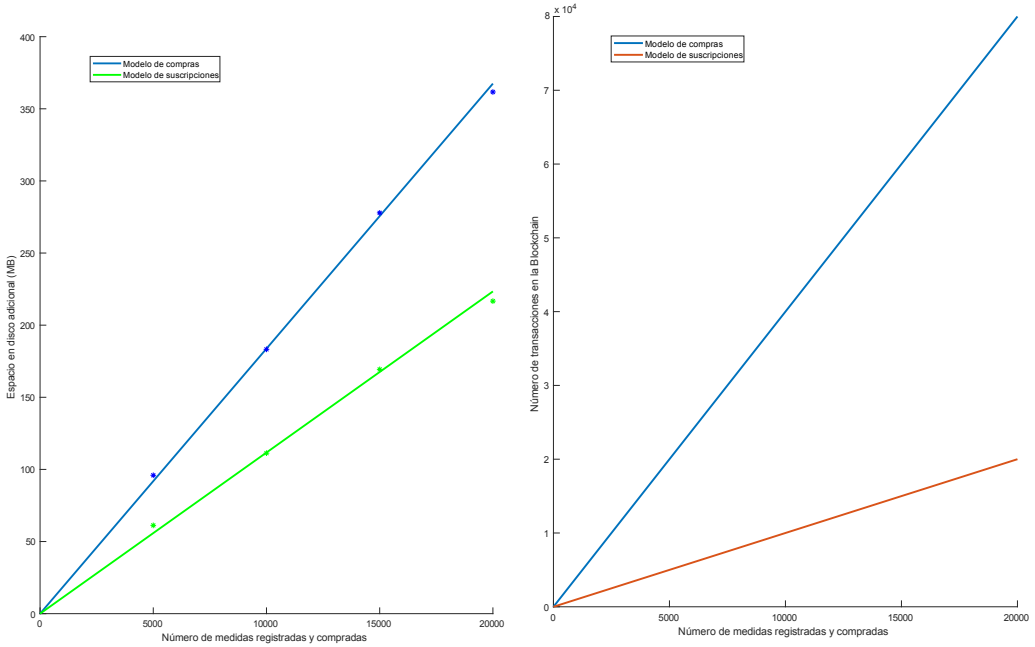
Hasta ahora se ha centrado el análisis en el tiempo que tardan ciertas operaciones o procesos en ejecutarse, pero esta no es la única métrica interesante para evaluar el comportamiento del BIDM a nivel computacional. Otro parámetro de gran importancia es el espacio en disco. Esto es, cuántos bytes adicionales se generan en el sistema de almacenamiento cada vez que tiene lugar una operación.

El proceso de medida se ha realizado tal como se explica a continuación. En primer lugar, se ha tomado nota del tamaño de las máquinas virtuales con una *Blockchain* vacía. Posteriormente, se han realizado 20000 registros de medidas, se han parado todos los procesos de la plataforma, y se ha vuelto a anotar el tamaño. Sobre la misma *Blockchain*, se han ejecutado 20000 compras, parado los procesos y tomado nota del tamaño una última vez. De esta forma, calculando la diferencia de tamaño entre cada uno de los tres instantes, se puede obtener una estimación del aumento de espacio ocupado en disco que provoca cada una de las operaciones. Al haber realizado un gran número de ellas, se puede calcular una media que hace que la estimación tenga suficiente precisión. Las Ecuaciones 5.1 y 5.2 muestran estos cálculos y su resultado.

$$\text{Registro de medidas: } \frac{427 \cdot 10^6 \text{ bytes}}{20000 \text{ registros}} = 21350 \frac{\text{bytes}}{\text{registro}} \tag{5.1}$$

$$\text{Compra de medidas: } \frac{305 \cdot 10^6 \text{ bytes}}{20000 \text{ compras}} = 15250 \frac{\text{bytes}}{\text{compra}} \tag{5.2}$$

Como se puede observar, el registro de medidas conlleva un aumento de espacio ocupado en disco de aproximadamente 21 KB, y es significativamente superior al proceso de compra, con unos 15 KB. Esta diferencia se debe a que el registro conlleva introducir una nueva medida en la base de datos y varios parámetros de la misma en la *Blockchain*, lo cual tiene una sobrecarga en comparación a la compra, que no introduce datos nuevos. Con estos datos, se puede hacer un análisis interesante que compara el modelo de compras y el modelo de suscripciones en términos de espacio. Es importante recordar que las suscripciones tienen la ventaja de que las adquisiciones de medidas son implícitas. Esto evita las múltiples transacciones en el proceso de compra, y provoca que el espacio en disco debido a las adquisiciones por parte de los consumidores sea 0. Dicho esto, se muestra en la Figura 5.12 la comparación en escalabilidad de ambos modelos, haciendo uso de los resultados de tamaño obtenidos anteriormente.



(a) Espacio en disco (b) Número de transacciones

Figura 5.12: Resultados de escalabilidad en espacio de disco.

El primer resultado que se muestra es la escalabilidad en términos de espacio en disco, tomado directamente en MB. Se trata de un modelo lineal que hace uso de los resultados obtenidos en los análisis anteriores, y muestra cómo la pendiente en el caso del modelo de compras es casi el doble que en el modelo de suscripciones. Como ya se ha comentado, esto se debe a que las compras en este último son implícitas y no provocan ninguna sobrecarga adicional. Además, se representan los puntos exactos que se han medido tras el registro y compra de 5000, 10000, 15000 y 20000 medidas, respectivamente, para respaldar experimentalmente que la escalabilidad presenta un comportamiento lineal. Un resultado diferente, aunque íntimamente relacionado, se muestra en la gráfica de la derecha. En este caso, se representa la escalabilidad en número de transacciones de la *Blockchain*. En capítulos previos se ha visto que los registros de medidas traen consigo una transacción, mientras que las compras requieren tres. Por tanto, la diferencia entre el modelo que requiere las operaciones de compra y el que las evita gracias al concepto de las suscripciones es sustancial. Como se puede observar, por cada registro y compra de una medida, el modelo de compras añade cuatro transacciones a la *Blockchain*, mientras que el modelo de suscripciones solo requiere una. Hay que tener en cuenta que la suscripción en sí genera una transacción, pero no escala con el número de medidas ya que solo se realiza una vez.

Con los resultados a la vista, se puede concluir que el modelo de suscripciones trae consigo una enorme ventaja respecto al modelo de compras individuales. Su escalabilidad, o en otras palabras, el espacio en disco que ocupa la plataforma crece de forma

mucho más lenta y por tanto permite la operatividad en dispositivos más limitados. En el siguiente apartado se va a realizar una comparación más extensa entre ambos modelos.

5.4.3 Comparativa

Una vez finalizada la evaluación de ambos modelos en términos de retardo y escalabilidad, se puede realizar una comparativa entre ellos en función de su rendimiento y funcionalidad. Como se ha visto, el modelo de suscripciones cuenta con una clara ventaja que nace del propio concepto de suscripción. El consumidor se interesa por un flujo de datos, y recibe implícitamente todas las medidas que pertenezcan a ese flujo. Esto beneficia significativamente la escalabilidad en tamaño del BIDM, al mismo tiempo que convierte el proceso de compra en una operación automática e instantánea. No obstante, a nivel de funcionalidad, el modelo de compras individuales sigue teniendo interés para consumidores potenciales que solo deseen la adquisición de medidas particulares. Por tanto, no se debe ver esta comparativa como un enfrentamiento entre ambos modelos para buscar cuál es superior, ya que realmente el BIDM cuenta con ambas posibilidades para dar la mayor flexibilidad posible a los consumidores. En este sentido, los dos modelos vistos trabajan en cooperación para dar el mejor servicio posible. Esta comparativa es, por consiguiente, una recopilación de las situaciones en las que el uso de un modelo u otro puede resultar beneficioso para el consumidor.

- **Modelo de compras individuales**
  - *Interés en medidas específicas*: El caso más claro de uso de este modelo es aquel que le da nombre. La compra de medidas individuales resulta interesante cuando el consumidor desea adquirir una medida específica. Este caso se puede dar en numerosas situaciones, por ejemplo, conocer la humedad en el instante actual medida por un sensor ubicado en un determinado lugar.
  - *Precio reducido*: Los productores IoT son quienes deciden el precio de sus medidas y suscripciones, pero es lógico suponer que una medida individual siempre va a tener un coste significativamente inferior a una suscripción, ya que esta última incluye múltiples medidas. Un consumidor que desee acceder a la funcionalidad de la plataforma haciendo un uso eficiente (o un uso mínimo) de capital encontrará más beneficioso realizar compras individuales.
  - *Adquisición instantánea*: Otra posible situación en la que se puede encontrar un consumidor es la necesidad de acceder a las medidas rápidamente. En una suscripción, el comprador ha de esperar a que el productor genere las medidas correspondientes al flujo al que se ha suscrito. Por el contrario, al comprar medidas individuales, el dato está disponible en el instante en el que se realiza la compra, si bien será una medida que se ha generado con anterioridad a la compra.

- *Medidas antiguas*: Como ya se ha comentado, la suscripción permite el acceso a todas las medidas de un determinado flujo que se generen con posterioridad a la suscripción. Sin embargo, con este mecanismo no es posible adquirir medidas pasadas. Si un consumidor desea acceder a estas medidas históricas, el modelo de compras individuales es la única opción posible.

- **Modelo de suscripciones**

  - *Interés en un tipo de medidas*: Al igual que en el modelo anterior, existe un caso de uso que es el principal objetivo de las suscripciones. Los compradores interesados en un flujo de medidas, como por ejemplo la temperatura en tiempo real en una determinada localización, se pueden suscribir a dicho flujo para recibir las medidas automáticamente en el momento en que se generan.
  - *Comodidad*: Una de las ventajas a nivel funcional del modelo de suscripciones es que, una vez realizada la suscripción, el comprador no tiene que hacer nada más que acceder a las medidas que desee. Sin embargo, en el modelo de compras individuales, el consumidor tiene que pasar por el proceso de compra cada vez que desea adquirir una nueva.
  - *Eficiencia*: Del mismo modo que en el caso anterior era lógico asumir que las medidas individuales serán más baratas que las suscripciones, también es de sentido común que los productores pongan las suscripciones a un precio tal que sean rentables para el consumidor a largo plazo. De lo contrario, cualquier cliente informado evitaría el uso de las suscripciones completamente. Por lo tanto, los compradores interesados en hacer un uso eficiente de los recursos económicos a largo plazo encontrarán que el modelo de suscripciones es la mejor opción.
  - *Programación y planificación*: En muchos casos, los compradores van a utilizar las medidas para realizar sus propias aplicaciones. La mejor forma de asegurar un flujo de datos consistente sin necesidad de la interacción del usuario es la suscripción. De esta forma, los compradores pueden programar y automatizar procesos que tengan las medidas a las que están suscritos como entrada.

Cabe mencionar que estos son casos de uso típicos y situaciones en las que puede resultar ventajoso el uso de uno de los modelos en concreto, pero los usuarios del BIDM tienen libertad para hacer uso de estos recursos de la forma en la que deseen. Esto hace que los casos de uso sean prácticamente ilimitados, ya que la única barrera es la imaginación de los usuarios que utilizan esta plataforma para desarrollar sus propias aplicaciones. Como se ha comentado, el objetivo de tener varios modelos en funcionamiento es que los clientes cuenten con la mayor flexibilidad posible. Por tanto, un modelo no sustituye al anterior aunque presente ventajas en términos de rendimiento.

Capítulo 6

Conclusiones y líneas futuras

En este trabajo se han implementado y analizado diversas funcionalidades en el BIDM. En primer lugar, se han realizado mejoras en el modelo de compras individuales y en el registro de medidas para minimizar el retardo de las operaciones manteniendo totalmente la funcionalidad original. Posteriormente, se ha concebido e implementado un nuevo modelo de suscripciones, que permite a los compradores adquirir automáticamente todas las medidas que pertenezcan a un mismo flujo de datos. Tras incluir sondas de medida a lo largo de las operaciones en ambos modelos, se ha realizado un análisis experimental exhaustivo de rendimiento alterando las condiciones de funcionamiento. Se ha completado la integración de *SmartSantander*, lo que valida el funcionamiento de la plataforma bajo condiciones reales, y se han desarrollado generadores sintéticos de medidas para dar una mayor profundidad al análisis. Finalmente, se han comparado ambos modelos tanto cuantitativamente, a nivel de rendimiento computacional, como cualitativamente. En este capítulo se van a resumir las conclusiones obtenidas, y comentar las líneas futuras que permiten continuar el desarrollo de este trabajo en diferentes ámbitos.

6.1 Conclusiones

En la Sección 2.2 se hizo una revisión del estado del arte que reveló algunas de las carencias que se observan en varios de los artículos relacionados con los mercados de datos que se revisaron. En este documento se ha tratado de realizar un análisis lo más completo y exhaustivo posible, justificando el uso de *Blockchain* como tecnología habilitadora para un mercado de datos. Este análisis intensivo supera el nivel de detalle que se puede observar en cualquier artículo publicado hasta hoy en esta línea de trabajo, y muestra el rendimiento computacional de la plataforma bajo numerosas condiciones. Además de analizar los procesos básicos de registro y compra de medidas, se han rediseñado y re-implementado los componentes de tal forma que tanto el retardo como la escalabilidad de estas operaciones se ha optimizado tanto como ha sido posible.

Para que los análisis sean rigurosos en un entorno real, también se ha integrado

la plataforma *SmartSantander* con el BIDM. De esta forma, se han podido utilizar medidas de sensores reales producidas por una infraestructura IoT real, comprobando su correcto funcionamiento y observando que su rendimiento no empeora respecto a las medidas generadas sintéticamente bajo diferentes distribuciones estadísticas. Numerosas automatizaciones han permitido realizar este análisis a gran escala y generar resultados con una alta verosimilitud.

Por otra parte, se ha extendido la operatividad de la plataforma con un nuevo modelo funcional. Con la apertura de una nueva posibilidad a la hora de adquirir medidas, el modelo de suscripciones a *datastreams*, los consumidores disfrutan de una gran flexibilidad a la hora de obtener los datos que desean para desarrollar sus propias aplicaciones, o darle el uso que consideren oportuno. Los productores también se benefician de ello, pues es un nuevo posible mercado al cual abrir su negocio de venta de medidas. Este nuevo modelo trabaja en tándem con la compra de medidas individuales para dar un servicio lo más amplio posible a los clientes. Tras incluir la escalabilidad de las suscripciones en el análisis, se ha observado que su comportamiento a gran escala es muy prometedor.

Por último, se resumen a continuación las conclusiones obtenidas de los análisis realizados en el Capítulo 5.

- La diferencia de retardo en las operaciones según estas se generan a mayor o menor velocidad es ínfima. En otras palabras, la plataforma es capaz de adaptarse a altos ritmos en la generación de operaciones por parte de los clientes (tanto productores como consumidores).

- La llegada de múltiples peticiones simultáneas provoca que existan bloques en la cadena que albergan más de una transacción. Esto es un punto a favor de la tecnología *Blockchain*, ya que queda demostrado que su funcionamiento interno se adapta sin problemas a situaciones de concurrencia de registros y/o compras.

- Al registrar medidas generadas por una plataforma IoT real, *SmartSantander*, el rendimiento del BIDM se mantiene invariable respecto a los experimentos realizados con medidas sintéticas. Por tanto, su funcionamiento y rendimiento están garantizados para entornos IoT reales.

- La longitud en bytes de las medidas registradas y compradas no afecta al retardo de las operaciones de manera apreciable.

- La escalabilidad en retardo de la plataforma es idónea: no se ha observado ninguna diferencia cuando la *Blockchain* tiene grandes cargas. Este experimento se ha realizado hasta los 500000 bloques, y si bien no se puede obviar la posibilidad de que la escalabilidad empeore tras un número mucho mayor, se puede suponer que esto no ocurre bajo condiciones de uso naturales.

- Buena parte del retardo del sistema se debe a los accesos a la base de datos, que son inevitables.

- La escalabilidad en espacio es un parámetro de alto interés, ya que en todos los casos el tamaño en disco de la *Blockchain* crece de manera lineal con el número de operaciones realizadas. Sin embargo, el modelo de *datastreams* presenta una pendiente mucho menor gracias a una mayor eficiencia en la generación de transacciones.

6.2 Líneas futuras

Tomando como punto de partida la implementación realizada en este trabajo, el BIDM aún tiene margen de mejora y nuevas funcionalidades que ofrecer. En particular, la plataforma aún no está orientada al negocio y carece de un sistema de monetización más allá de los *tokens* necesarios para realizar compras y suscripciones. Hay unos pocos ejemplos en la literatura que tratan de establecer modelos de negocio, o sistemas de *reviews* en los que los consumidores valoran las medidas o a los productores IoT. Es posible integrar un sistema de pago real, o cambiar totalmente el mecanismo de consenso para poder utilizar el *Ether* directamente como moneda. Sin embargo, esta última opción tendría implicaciones sobre el funcionamiento del BIDM desde su núcleo.

Otra de las líneas futuras más interesantes es la creación de un sistema de control de uso, o *Data Usage Control*. Este modelo consiste en restringir el uso que los consumidores de medidas hacen de las mismas, de forma este uso está limitado a una serie de opciones. Con ello se pretende ofrecer una funcionalidad mediante la cual los dueños de la información tengan la capacidad de trazar todo el ciclo de vida de la información generada por la infraestructura IoT de la que son propietarios. Para conseguir este efecto y que los accesos a cada funcionalidad sean transparentes e inmutables, es posible utilizar un contrato inteligente. Cuando un consumidor adquiera una medida, en lugar de tener acceso directo a ella, puede escoger entre una serie de opciones validadas por el BIDM. Cada vez que se utiliza una de estas opciones, queda plasmado en una transacción.

Por otra parte, el despliegue de la plataforma sobre sistemas con especificaciones más reducidas, como una *Raspberry Pi*, podría aclarar nuevos detalles sobre su rendimiento bajo condiciones limitadas.

Otras posibilidades incluyen la conexión con la *Blockchain* pública de *Ethereum*, el despliegue de los componentes sobre contenedores *Docker* para mayor portabilidad, o la modernización de la interfaz web con un diseño atractivo para los clientes.

Bibliografía

[1] *Bitcoin*. https://bitcoin.org/es/. Accedido: 07-08-2021.

[2] *Ethereum*. https://ethereum.org/es/. Accedido: 07-08-2021.

[3] Iván González. "Desarrollo de un Marketplace para la Internet de las Cosas basado en Blockchain". En: sep. de 2020.

[4] *Qué es una cadena de bloques*. https://www.criptonoticias.com/criptopedia/que-es-una-cadena-de-bloques-block-chain/. Accedido: 07-08-2021.

[5] A. Narayanan y col. "Bitcoin and cryptocurrency technologies: a comprehensive introduction". En: mayo de 2016. DOI: OCLC-1145890545.

[6] *Go-Ethereum*. https://geth.ethereum.org/. Accedido: 07-08-2021.

[7] *IPFS*. https://ipfs.io/. Accedido: 07-08-2021.

[8] *Cloudflare anuncia el soporte a la tecnología IPFS*. https://tecnonucleous.com/2018/09/19/ipfs-cloudflare/. Accedido: 07-08-2021.

[9] D. Wang y col. "From IoT to 5G I-IoT: The Next Generation IoT-Based Intelligent Algorithms and 5G Technologies". En: *2018 IEEE Communications Magazine*. Vol. 56. 10. Oct. de 2018, págs. 114-120. DOI: 10.1109/MCOM.2018.1701310.

[10] A. Lee y col. "A Hybrid Software Defined Networking Architecture for Next-Generation IoTs". En: *KSII Transactions on Internet and Information Systems 2018*. 12. 2018, págs. 932-945. DOI: 10.3837/tiis.2018.02.024.

[11] Xiangchen Zhao y col. "Demo Abstract: The Intelligent IoT Integrator Data Marketplace - Version 1". En: *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*. 2020, págs. 270-271. DOI: 10.1109/IoTDI49375.2020.00042.

[12] Anish Agarwal, Munther A. Dahleh y Tuhin Sarkar. "A Marketplace for Data: An Algorithmic Solution". En: *CoRR* abs/1805.08125 (2018). arXiv: 1805.08125. URL: http://arxiv.org/abs/1805.08125.

[13] Www Org y col. "The Data Marketplace Survey Revisited". En: (mar. de 2014).

[14] Florian Stahl, Fabian Schomm y Gottfried Vossen. "Data marketplaces: An emerging species". En: *Frontiers in Artificial Intelligence and Applications* (ene. de 2014), págs. 145-158. DOI: 10.3233/978-1-61499-458-9-145.

[15] Ji-Sun Park y col. “Smart Contract-Based Review System for an IoT Data Marketplace”. En: *Sensors* 18.10 (2018). ISSN: 1424-8220. DOI: 10.3390/s18103577. URL: https://www.mdpi.com/1424-8220/18/10/3577.

[16] Nick Hynes y col. “A Demonstration of Sterling: A Privacy-Preserving Data Marketplace”. En: *Proc. VLDB Endow.* 11.12 (ago. de 2018), 2086–2089. ISSN: 2150-8097. DOI: 10.14778/3229863.3236266. URL: https://doi.org/10.14778/3229863.3236266.

[17] Pooja Gupta, Salil S. Kanhere y Raja Jurdak. “A Decentralized IoT Data Marketplace”. En: *CoRR* abs/1906.01799 (2019). arXiv: 1906.01799. URL: http://arxiv.org/abs/1906.01799.

[18] Hyunkyung Yoo y Namseok Ko. “Blockchain based Data Marketplace System”. En: *2020 International Conference on Information and Communication Technology Convergence (ICTC)*. 2020, págs. 1255-1257. DOI: 10.1109/ICTC49870.2020.9289087.

[19] Prabal Banerjee y Sushmita Ruj. “Blockchain Enabled Data Marketplace - Design and Challenges”. En: *CoRR* abs/1811.11462 (2018). arXiv: 1811.11462. URL: http://arxiv.org/abs/1811.11462.

[20] Sebastian Lawrenz, Priyanka Sharma y Andreas Rausch. “Blockchain Technology as an Approach for Data Marketplaces”. En: *Proceedings of the 2019 International Conference on Blockchain Technology*. ICBCT 2019. New York, NY, USA: Association for Computing Machinery, 2019, 55–59. ISBN: 9781450362689. DOI: 10.1145/3320154.3320165. URL: https://doi.org/10.1145/3320154.3320165.

[21] Konstantinos Christidis y Michael Devetsikiotis. “Blockchains and Smart Contracts for the Internet of Things”. En: *IEEE Access* 4 (2016), págs. 2292-2303. DOI: 10.1109/ACCESS.2016.2566339.

[22] Krešimir Mišura y Mario Žagar. “Data marketplace for Internet of Things”. En: *2016 International Conference on Smart Systems and Technologies (SST)*. 2016, págs. 255-260. DOI: 10.1109/SST.2016.7765669.

[23] Pooja Gupta y col. “Towards a blockchain powered IoT data marketplace”. En: *2021 International Conference on COMmunication Systems NETworkS (COMSNETS)*. 2021, págs. 366-368. DOI: 10.1109/COMSNETS51098.2021.9352865.

[24] Gowri Sankar Ramachandran, Rahul Radhakrishnan y Bhaskar Krishnamachari. “Towards a Decentralized Data Marketplace for Smart Cities”. En: *2018 IEEE International Smart Cities Conference (ISC2)*. 2018, págs. 1-8. DOI: 10.1109/ISC2.2018.8656952.

[25] Miyeon Ha y col. "Where WTS meets WTB: A Blockchain-based Marketplace for Digital Me to trade users' private data". En: *Pervasive and Mobile Computing* 59 (2019), pág. 101078. ISSN: 1574-1192. DOI: https://doi.org/10.1016/j.pmcj.2019.101078. URL: https://www.sciencedirect.com/science/article/pii/S1574119218305947.

[26] Ronghua Xu y col. "BlendSM-DDM: BLockchain-ENabled Secure Microservices for Decentralized Data Marketplaces". En: *2019 IEEE International Smart Cities Conference (ISC2)*. 2019, págs. 14-17. DOI: 10.1109/ISC246665.2019.9071766.

[27] Ahmad Alsharif y Mahmoud Nabil. "A Blockchain-based Medical Data Marketplace with Trustless Fair Exchange and Access Control". En: *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 2020, págs. 1-6. DOI: 10.1109/GLOBECOM42002.2020.9348192.

[28] Vlasis Koutsos y col. "Agora: A Privacy-aware Data Marketplace". En: *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. 2020, págs. 1211-1212. DOI: 10.1109/ICDCS47774.2020.00156.

[29] Wiem Badreddine, Kaiwen Zhang y Chamseddine Talhi. "Monetization using Blockchains for IoT Data Marketplace". En: *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. 2020, págs. 1-9. DOI: 10.1109/ICBC48266.2020.9169424.

[30] *NodeJS*. https://nodejs.org. Accedido: 07-08-2021.

[31] *API Smart Santander*. https://api-dev.smartsantander.eu:10443/. Accedido: 27-08-2021.