Universidad Zaragoza
1542

Trabajo Fin de Máster

Fiabilización de la carta de propiedad de un vehículo y su aplicación a la red de postventa mediante la tecnología blockchain.

Reliability of the certificate of ownership of a vehicle and its application to the after-sales network through blockchain technology.

Autor

Javier Fernández Domínguez

Director

Antonio Crespo Ramos

Titulación del autor

Máster en ingeniería industrial

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2021

Resumen

La tecnología blockchain inició su recorrido en 2008 de la mano de Bitcoin, una moneda virtual a la que esta tecnología le proporciona unas características únicas: descentralización, transparencia, seguridad y fiabilidad.

Con el paso de los años, el concepto de blockchain ha ido evolucionando, dando lugar a cadenas de bloques más complejas, como Ethereum, las cuales permiten la incorporación de programas automatizados y el aseguramiento de mayor diversidad de datos gracias a los Smart contracts.

Aprovechando estas características únicas, este trabajo pretende demostrar la aplicabilidad de la tecnología blockchain para ofrecer servicios y gestionar la propiedad de los usuarios, centrándose en el desarrollo de una serie de servicios de postventa para vehículos.

Para ello se creará un certificado de propiedad utilizando tokens NTF, los cuales servirán de base para ofrecer un servicio de compraventa de vehículos, un servicio de alquiler y la recopilación de información durante la vida del vehículo en un pasaporte digital. También se estudiará la forma de introducir estas ideas en el mercado de automóviles planteando un modelo de negocio.

La tecnología demuestra ser prometedora y ofrece la posibilidad de extender su uso a una gran multitud de ámbitos, sin embargo, se debe seguir trabajando en la escalabilidad de las cadenas de bloques para que pueda generalizarse su aplicación.

Abstract

Blockchain technology began its journey in 2008 with Bitcoin, a virtual currency to which this technology provides unique characteristics: decentralization, transparency, security and reliability.

Over the years, the concept of blockchain has evolved, giving rise to more complex blockchains, such as Ethereum, which allow the incorporation of automated programs and the assurance of greater diversity of data thanks to Smart contracts.

Taking advantage of these unique characteristics, this work aims to demonstrate the applicability of blockchain technology to offer services and manage user ownership, focusing on the development of a series of after-sales services for vehicles.

To do this, a certificate of ownership will be created using NTF tokens, which will serve as the basis for offering a vehicle sale service, a rental service and the collection of information during the life of the vehicle in a digital passport. It will also be studied how to introduce these ideas in the car market by raising a business model.

The technology proves promising and offers the possibility of extending its use to a large multitude of areas, however, work must continue on the scalability of blockchains so that their application can be generalized.

Contenido

1. La tecnología blockchain

1.1. La tecnología Blockchain

La tecnología blockchain, o cadena de bloques en español, consiste en un libro contable descentralizado, digitalizado y distribuido, en el que se almacena información que, gracias a técnicas criptográficas, es resistente a la manipulación. Los nuevos datos se añaden en forma de conjuntos de información llamados bloques, los cuales se enlazan con los anteriores formando una cadena. Para poderse añadir un nuevo bloque a la cadena, se debe seguir un protocolo de consenso en el que el nuevo bloque, es verificado por los ordenadores participantes en la red. Al ser un sistema distribuido y descentralizado, no solo existe una copia de la blockchain, sino muchas a través de toda la red. [1]

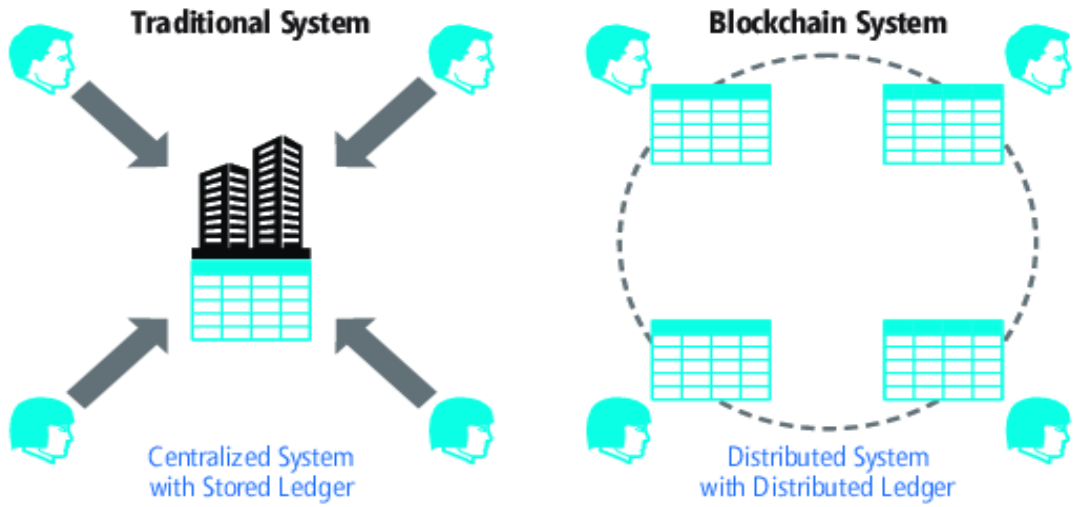

Figura 1. Libro contable centralizado y distribuido. Fuente: Microsoft [2]

La tecnología blockchain se ha popularizado durante la última década gracias al surgimiento del bitcoin en 2008, cuando una persona anónima bajo el pseudónimo Satoshi Nakamoto publicó el white paper "*Bitcoin: un sistema de dinero en efectivo electrónico peer-to-peer*" donde se describía el funcionamiento de una divisa digital basada en esta técnica. [3]

Sin embargo, ya desde los comienzos de internet se fueron dando una serie de avances en criptografía de clave pública, cadenas de bloques, redes peer to peer y sellado de tiempo, que terminaron convergiendo en el bitcoin.

En 1991 Stuart Haber y W. Scott Stornetta firman el primer trabajo de cadena de bloques segura utilizando criptografía. En 1998, Wei Dai describe una solución descentralizada para pagos electrónicos basada en criptografía de clave pública. Sobre estos trabajos se continúa evolucionando la técnica, hasta la llegada de bitcoin y las arquitecturas de blockchain más recientes. [4] [5] [6]

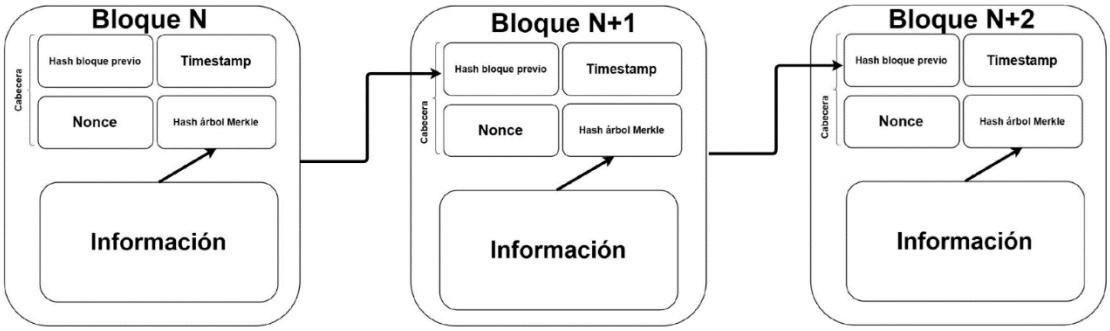

Figura 2. Estructura de los bloques de la blockchain. [7]

1.1.1. Blockchain vs libro contable distribuido

Se ha comentado anteriormente que el blockchain consiste en un libro contable distribuido. Sin embargo, no todos los libros contables distribuidos son blockchains.

Un libro contable distribuido es una base de datos que existe en diferentes localizaciones y que poseen múltiples participantes. Muchas compañías utilizan bases de datos centralizadas, en las que existiría un solo punto de fallo. Para evitar esto, hay empresas que utilizan un sistema descentralizado para procesar, validar y autentificar transacciones e intercambios de datos. Todos estos archivos tienen una marca de tiempo y se les da una firma criptográfica única cuando los participantes han llegado a un consenso.

Una cadena de bloques es un libro contable distribuido formado por una serie de paquetes o bloques de información que se van añadiendo a lo largo del tiempo y tienen una dependencia lógica con el anterior. Esta dependencia lógica se construye con una función criptográfica llamada hash. [8]

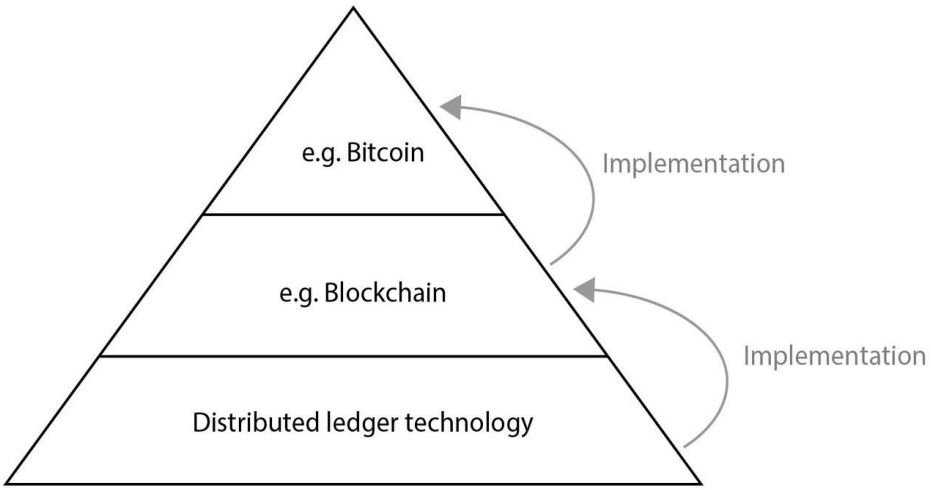

Figura 3. Diferencia entre libro contable y blockchain. [9]

1.1.2. Blockchain pública vs blockchain privada

Las cadenas de bloques se pueden clasificar en función del acceso a los datos almacenados.

Las blockchains públicas permiten que cualquiera se pueda crear una dirección e interactuar con ella. Son una forma interesante para las empresas de interactuar con los clientes o de clientes con otros clientes. Ejemplos de este tipo de cadenas son Bitcoin o Ethereum.

Sus características principales son:

- Descentralización: Los sistemas públicos deben ser descentralizados y distribuidos, de forma que ninguna entidad (o gobierno) puede cerrar o censurar la red.
- Anonimato: En las blockchains públicas, los participantes pueden mantenerse anónimos si así lo desean. Este puede ser un parámetro deseable en algunas situaciones y problemático en otras.
- Escalabilidad: El proceso de validación es costoso y existen problemas de escalabilidad que pretenden ser resueltos a medida que evolucione la tecnología.

Una blockchain privada forma un ecosistema cerrado en el que cada participante en la red está bien definido. Este tipo de blockchain permite que organizaciones o consorcios intercambien información de una forma eficiente. Solo las entidades previamente aceptadas pueden validar bloques y acceder a la información. Ejemplos de este tipo de cadenas son las ofrecidas por Hyperledger Fabric o Corda.

Las características de descentralización, privacidad y transparencia dependerán de cómo los miembros del consorcio deseen organizarse entre ellos. Este tipo de blockchain es más escalable que la pública debido a que, al tener un control de los participantes, no son necesarios mecanismos de consenso tan intensivos. [10]

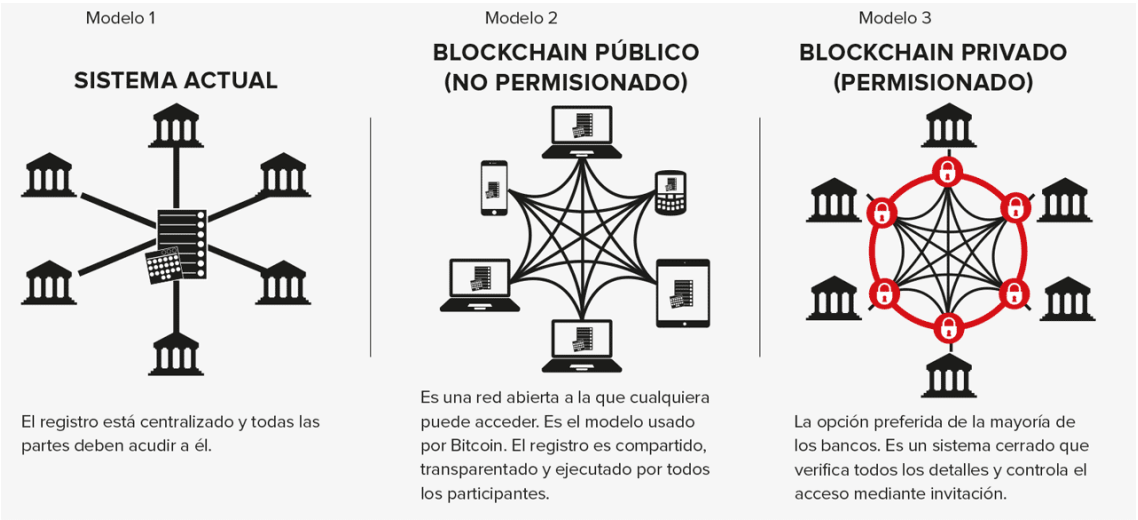

Figura 4. Modelos de blockchain. [11]

1.2. La tecnología detrás del blockchain

1.2.1. La función hash

Una función hash es un procedimiento criptográfico donde se emplea un algoritmo específico (SHA-256 para Bitcoin y Keccak-256 para Ethereum) para transformar una información de entrada (P.ej. un texto) en una secuencia alfanumérica única de **longitud fija**, denominada hash. [12]

El hash es una función unidireccional, no encripta la información, ya que no es posible obtener el mensaje original a partir del hash. Esto es debido a que en el proceso de cálculo del hash existe una pérdida de información.

Solo sería posible obtener el texto original de un hash mediante un ataque de fuerza bruta, en el que se probasen posibles entradas viables y comprobando si el hash calculado es el mismo.

Es una función determinista, es decir, que para una misma entrada, la salida siempre será la misma. Sin embargo, si se produce un mínimo cambio en la información de entrada, el resultado de la operación será completamente diferente.

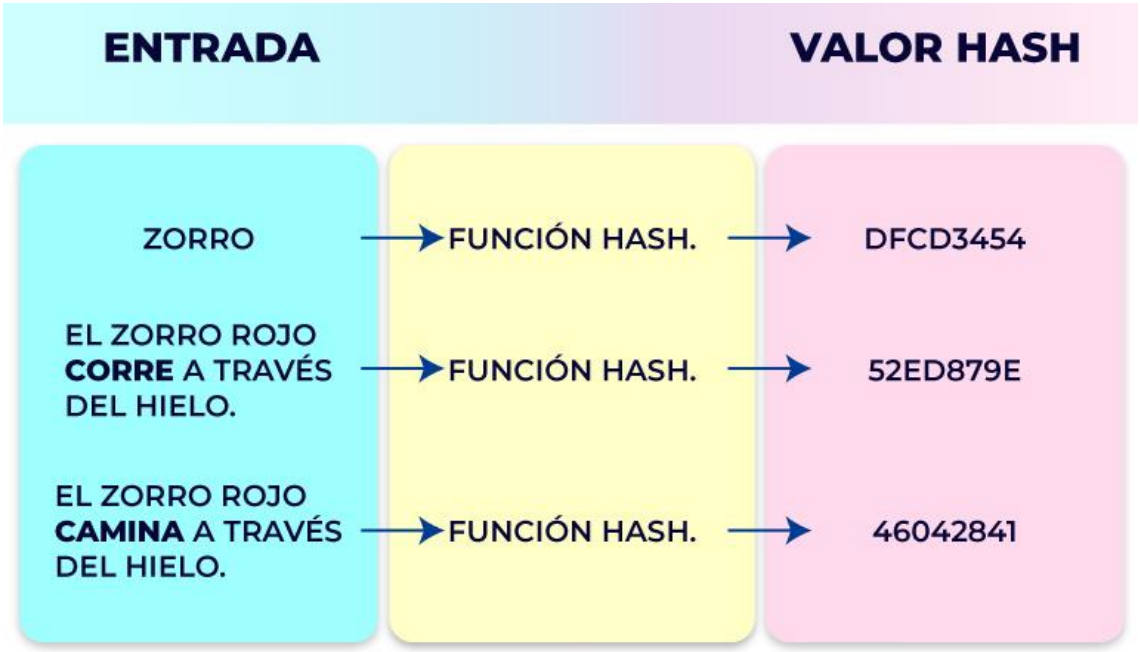

Figura 5. Una función hash en funcionamiento. Fuente: Wikipedia

Como se ha comentado, existe pérdida de información en el cálculo de la función, esto implica que existe la posibilidad de que para dos entradas diferentes se obtenga el mismo hash. Esta situación se denomina "colisión" y para que la función hash sea segura, la probabilidad de que esto ocurra debe ser prácticamente cero.

La función hash SHA-256, tiene 2^{256} salidas posibles, por lo que para tener posibilidades relevantes de conseguir dos hashes iguales sería necesario calcular el hash de una cantidad cercana de entradas. Como comparativa, si en condiciones ideales pudiésemos capturar toda la

energía del Sol durante 32 años y utilizarla para computar hashes, solo conseguiríamos probar 2^{192} veces. [13]

En Bitcoin la función hash se utiliza en tres áreas: la creación de la dirección, la minería de bloques mediante el protocolo proof of work y para estandarizar la longitud de la información que se firma digitalmente.

1.2.2. Firmas digitales asimétricas

Una firma digital es un mecanismo criptográfico que permite al receptor de un mensaje firmado digitalmente identificar al emisor de éste (autentificación) y confirmar que el mensaje no ha sido alterado desde que fue firmado (integridad).

Dentro de los diferentes esquemas de firmas digitales, en blockchain se utiliza criptografía asimétrica, también llamada criptografía de clave pública. Esta consiste en el intercambio de información utilizando dos claves matemáticamente relacionadas, una de ellas es pública y se puede compartir, mientras que la otra es privada y es la que se utiliza para firmar el mensaje.

El esquema básico es el siguiente [14]

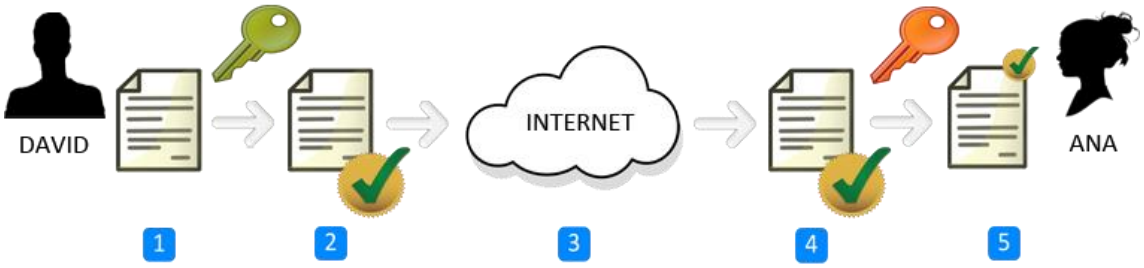

Figura 6. Esquema de firma. Fuente: Wikipedia

1- David redacta un mensaje.
2- David firma el mensaje con su **clave privada**.
3- David envía el mensaje firmado digitalmente a Ana a través de internet.
4- Ana recibe el mensaje firmado digitalmente y comprueba su autenticidad usando la **clave pública** de David.
5- Ana ya puede leer el mensaje con total seguridad de que ha sido David el remitente.

El proceso seguido en Bitcoin es el siguiente:

Primero se realiza un hash de la información para estandarizar la longitud de los datos. Tras ello, se firma el mensaje utilizando una clave privada. Hay que señalar que las firmas digitales están directamente relacionadas con el contenido del mensaje, por lo que cada una será diferente. Una vez el mensaje es enviado para ser incluido en la blockchain, se puede verificar la validez de la firma utilizando la clave pública del emisor.

1.2.3. Criptografía de curva elíptica

¿Cómo es posible que exista esta relación entre clave pública, clave privada y firma? La respuesta está en la criptografía de curva elíptica.

La criptografía de curva elíptica es una variante de la criptografía asimétrica basada en matemáticas de curvas elípticas. El sistema se basa en la dificultad de encontrar la solución a ciertos problemas matemáticos. Uno de ellos es el logaritmo discreto. Encontrar b dada la ecuación $a^b = c$ siendo a y c conocidos tiene una complejidad exponencial, mientras que resolver el problema inverso es muy sencillo. Haciendo uso de curvas elípticas se incrementa la complejidad de resolver el problema. [15]

La curva elíptica [16] del protocolo de bitcoin es $y^2 = x^3 + 7$

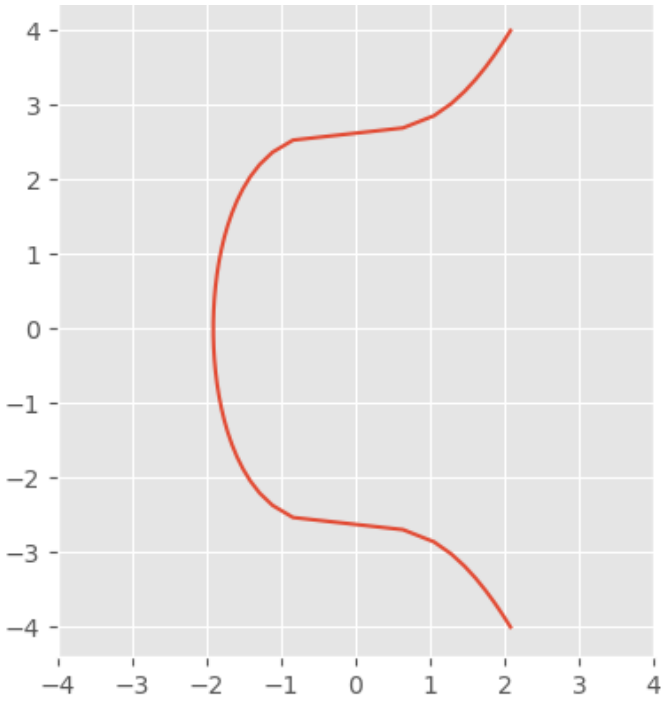

Figura 7. Curva elíptica utilizada por Bitcoin (Secp256k1).

Una forma de aplicar este sistema es ECDSA, un algoritmo que emplea operaciones sobre puntos de curvas elípticas para brindar seguridad. El proceso es el siguiente [17]:

Generación de las claves:

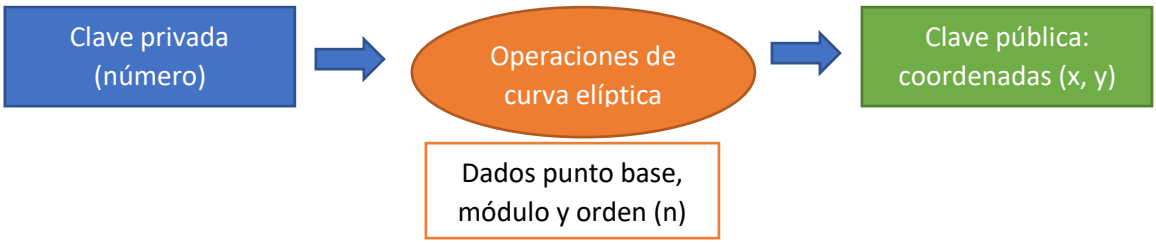

Figura 8. Generación de la pareja de claves: pública y privada. Fuente: Elaboración propia

Proceso de firma:

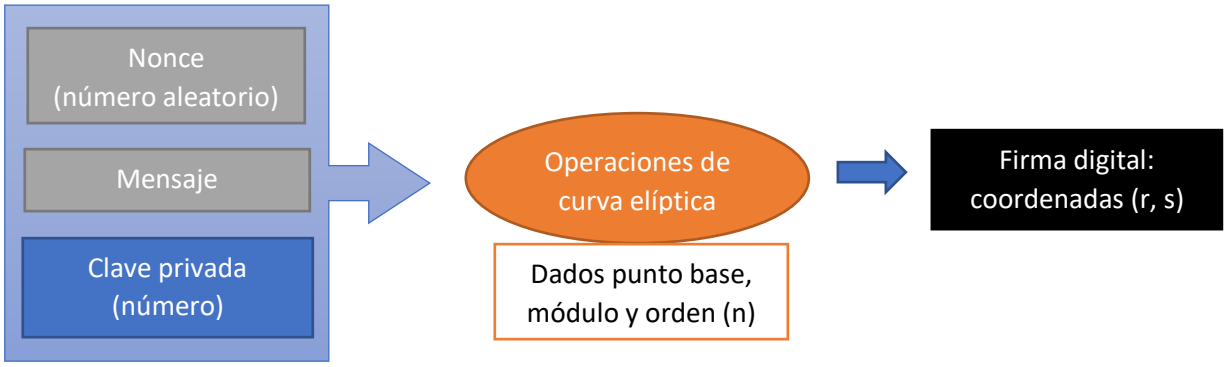

Figura 9. Proceso de firma. Fuente: Elaboración propia

Proceso de verificación:

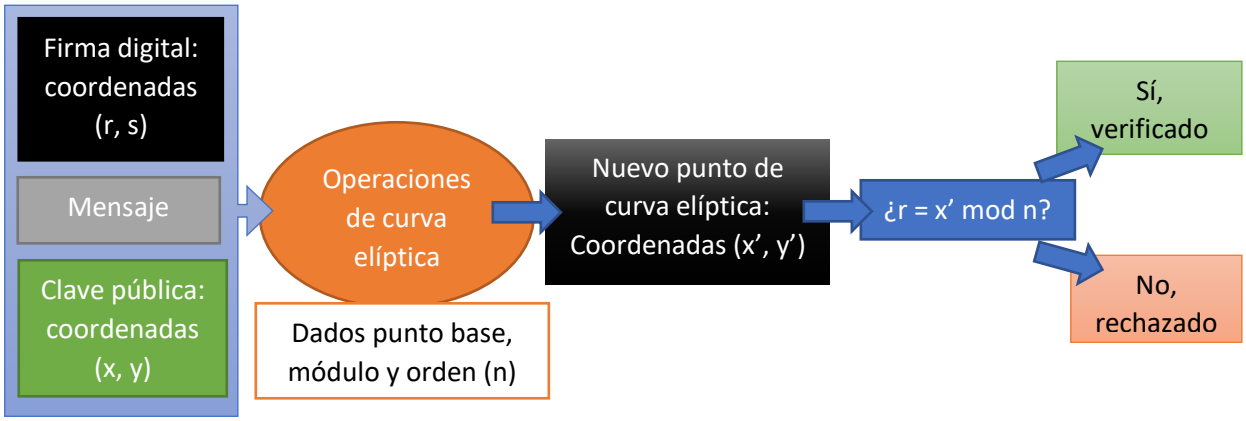

Figura 10. Proceso de verificación. Fuente: Elaboración propia

1.2.4. Algoritmos de consenso

Los algoritmos de consenso son elementos cruciales de todas las redes blockchain, debido a que son responsables de mantener la integridad y seguridad. Uno de los principales problemas de una red distribuida, es llegar a un consenso único entre todas sus partes. Al no existir una autoridad central, sus nodos necesitan ponerse de acuerdo respecto a la validez de las transacciones. [18] [19]

1.2.4.1. Proof of Work (PoW)

Proof of work o prueba de trabajo es el algoritmo de consenso que utilizan Bitcoin, Ethereum y otras blockchains. Su esencia reside en que sólo después de que un nodo demuestre un trabajo computacional elevado, un bloque puede ser añadido a la blockchain, siempre que, las transacciones que contiene sean válidas.

El proceso de validación es el siguiente:

1. Recolección de transacciones

Un nodo recoge las transacciones que son emitidas a toda la red y las almacena en un bloque.

El nodo debe verificar que las transacciones sean válidas. Si no lo fuesen, el bloque sería rechazado por el resto de los nodos (recordemos que esto se puede hacer debido al sistema de firmas digitales).

Las transacciones pagan una comisión al nodo para ser incluidas en el bloque. A mayor comisión, antes es incluida en el bloque. Si la comisión es demasiado baja, se corre el riesgo de que nunca sea agregada. Esto se hace debido a que el tamaño de un bloque es finito y solo puede contener una cantidad limitada de información.

2. Minado del bloque

Comienza el minado del bloque. Para ello, el nodo debe resolver un problema muy complejo de generación de hashes con las transacciones como input del problema.

En el caso de Bitcoin, el problema consiste en realizar el hash del encabezado del bloque, el cual, como requisito, debe comenzar con cierta cantidad de ceros. El encabezado consta de los siguientes parámetros

- El hash del bloque anterior: Esto asegura que el bloque que va a ser minado es descendencia del bloque previo.
- Raíz de Merkle: Es un “hash de hashes” de todas las transacciones incluidas en el bloque. Esto permite simplificar el proceso de verificación del bloque, ya que, de esta forma, no es necesario comprobar todos los bytes del bloque, al mismo tiempo que quedan relacionados de forma segura con esta raíz.
- Nonce: Es el contador que el minero irá incrementando mientras prueba hashes hasta que uno dé la casualidad de ser válido y comenzar con la cantidad de ceros requeridos. (Se recuerda que este pequeño cambio es suficiente para que cada hash calculado sea completamente diferente del anterior y, por tanto, el proceso sea aleatorio).
- Marca de tiempo: También contribuye a dar variabilidad a los intentos de cálculo del hash.
- Otros datos: Como la versión del software, el número de bloque y la dificultad objetivo (número de ceros requeridos en el hash).

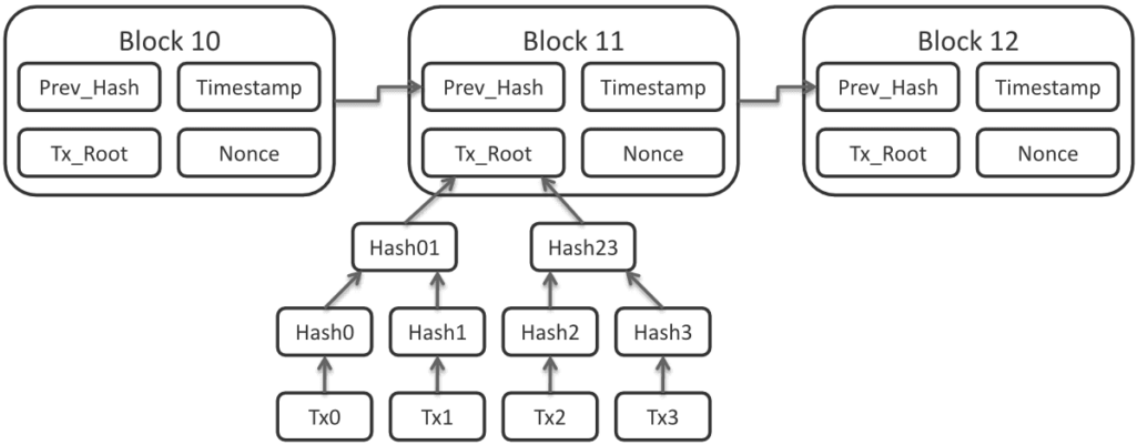

Figura 11. Esquema de la información de los bloques de Bitcoin. Fuente: Wikipedia

El número de ceros necesario para que un hash sea válido depende de la velocidad a la que se está consiguiendo resolver el problema y se adapta para que el tiempo de minado entre bloques sea aproximadamente de 10 minutos, en el caso de Bitcoin. En otras blockchains es diferente, siendo 15 segundos para Ethereum, 2,5 minutos para Litecoin o 1 minuto para Dogecoin. [20] [21]

La motivación para que un minero realice este esfuerzo computacional es recibir como premio criptomonedas de nueva creación y las comisiones de las transacciones.

3. Transmisión, verificación y confirmación

Una vez un minero ha conseguido un hash válido, emite el bloque a la red. El resto de los participantes verifican que el hash y que las transacciones son correctas. Tras ello, se añade a la blockchain y se tomará como punto de partida para la generación de un nuevo bloque.

El problema del doble gasto

Éste es el principal problema que aborda el algoritmo de Proof of Work.

Supongamos que un participante malicioso gasta sus bitcoins y los envía a una dirección a cambio de algo. Al mismo tiempo, realiza una operación paralela en la que realiza ese mismo gasto, pero intercambiando productos con otra persona.

¿Cuál se ejecutaría?

Pues si ambas se intentan añadir al mismo bloque, la segunda que se haya recibido sería rechazada, ya que no se contarían con fondos suficientes y se invalidaría.

¿Pero si cada una se incluye en diferentes bloques?

En este caso la operación aceptada sería la que pertenece al bloque minado (el que consigue un hash que comience por x ceros).

¿Y si el bloque en el que está recogida la operación paralela se sigue intentando minar hasta que se obtiene también un hash que comience por x número de ceros?

En este caso, esta cadena paralela iría con cierto retraso respecto a la anterior (porque ha tardado más tiempo en conseguir el hash válido), y el algoritmo de consenso requiere tomar como cadena válida *la cadena existente más larga*. Esto es muy importante, ya que, un actor malicioso no puede incluir una transacción en un bloque y acto seguido, dar un paso atrás y generar una nueva cadena paralela en la que realiza transacciones diferentes. Esto es porque ningún participante honrado continuaría trabajando sobre la cadena paralela, más corta.

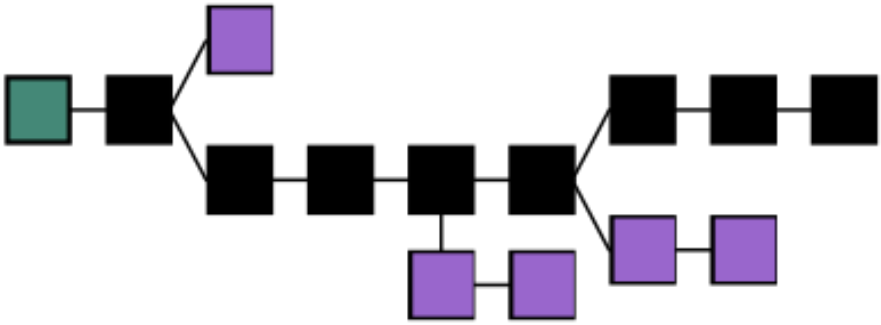

Figura 12. Cadena de bloques. Los bloques de la cadena más larga (negros) son los reconocidos como válidos. Las bifurcaciones (morado) quedan huérfanas.

¿Podría el actor malicioso seguir trabajando sobre la cadena paralela hasta alcanzar a la cadena más larga?

El actor malicioso podría seguir intentando trabajar sobre la cadena más corta hasta intentar alcanzar y superar a la larga. Sin embargo, hacer esto sería un trabajo inútil, ya que, al requerirse un enorme trabajo computacional para obtener un hash válido, los participantes honrados (que son muchos más) ganarían más y más distancia en número de bloques añadidos a la cadena. Es posible que, por azar (porque consigue 2 o 3 hashes válidos extremadamente rápido), el atacante malicioso consiga superar con su segunda cadena a la primera. Esta probabilidad decrece exponencialmente a medida que se añaden nuevos bloques tras el bloque de la transacción. Por ello, se recomienda esperar a que se añadan 3 o 4 bloques más detrás, para tener la seguridad de que una transacción es firme.

Ataque del 51%

La situación descrita en el párrafo anterior deja de ser válida si el actor malicioso consigue acaparar el 51% de la potencia computacional sobre la blockchain. En esta situación sí tendría la capacidad de reescribir una nueva cadena cambiando las transacciones hasta superar la que están tomando como válida el resto de los participantes.

Una situación así es menos probable a medida que más usuarios participan en la blockchain. Además, los incentivos para hacer esto son negativos ya que, al hacerse, se acabaría con la confianza en la blockchain y, como consecuencia, su valor monetario desaparecería. Esta situación, a nadie que haya realizado una inversión tan grande como para acaparar tanta potencia de cómputo, le interesa que ocurra, a no ser que su único fin sea destruir la blockchain.

Ataque del 51% (doble – gasto)

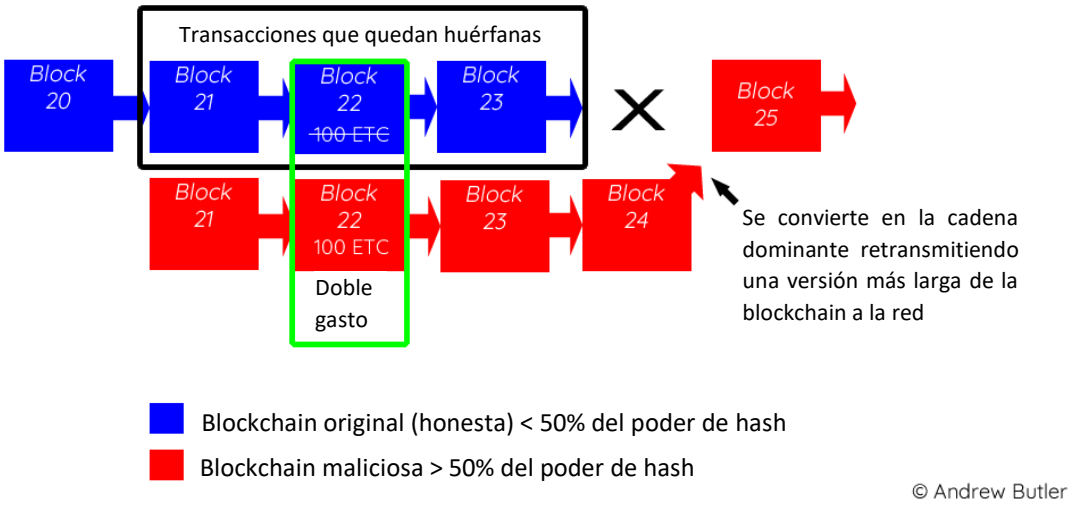

Figura 13. Esquema de un ataque del 51%. [22]

Fortalezas del algoritmo PoW

- Ha demostrado ser confiable y robusto
- Se puede predecir fácilmente el tiempo entre bloques.
- La única vulnerabilidad conocida es el "ataque del 51%".
- No se puede censurar

Debilidades del algoritmo PoW

- El gasto energético es enorme. La competición entre mineros para conseguir un hash válido resulta en un gran consumo de recursos.
- La alta competencia en el minado hace que sólo grandes pools de minería, con equipos especializados, estén en disposición de conseguir minar un bloque, lo que termina llevando a una centralización de recursos.
- Problemas de escalabilidad. [23]

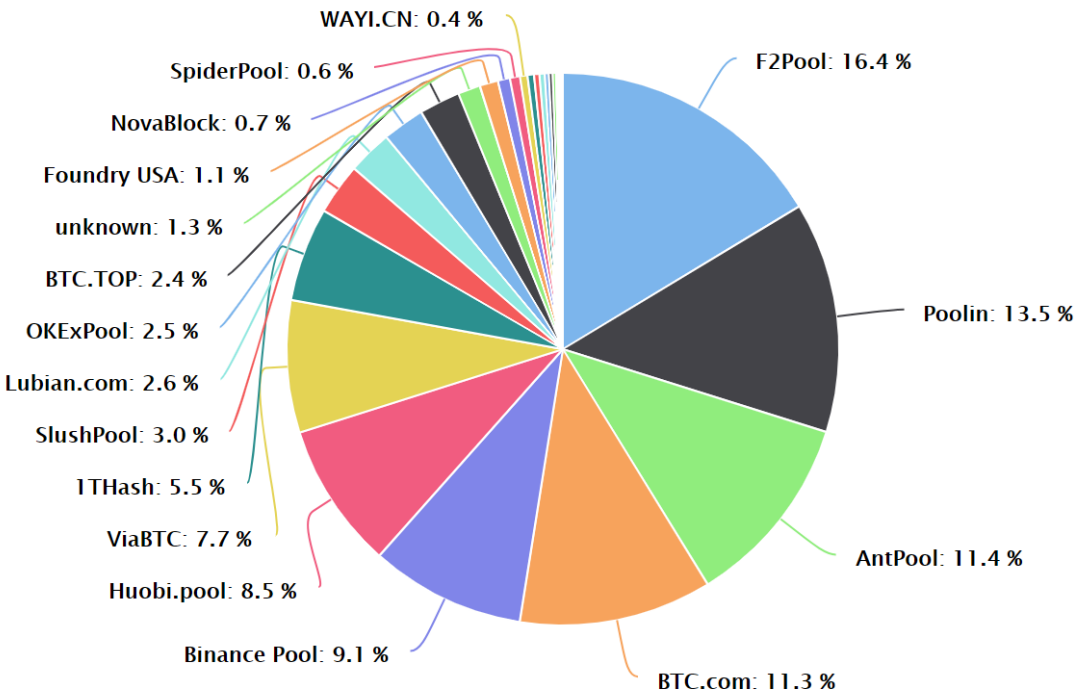

Figura 14. Porcentaje de la potencia de cómputo generada por los principales pools de minería. Los cuatro más importantes suman más del 51%. [24]

1.2.4.2. Proof of Stake (PoS)

Proof of Stake o prueba de participación, es un algoritmo de consenso en el que la decisión sobre qué nodo va a validar el próximo bloque se hace de forma aleatoria, pero dando mayores probabilidades a quienes cumplan una serie de criterios. El criterio más comúnmente utilizado es favorecer a quien aporte mayor cantidad de monedas como garantía.

Este modelo evita tener que realizar el gran trabajo computacional requerido en PoW y, por tanto, no se necesita consumir mucha energía ni tener hardware especializado.

Para realizar un ataque del 51% se necesitaría que alguien consiguiera acaparar el 51% de todas las monedas de la red, lo cual es menos probable y llega a ser prácticamente imposible a medida que el valor monetario de la blockchain aumenta. Además, realizar el ataque conllevaría la destrucción del valor de la red y el atacante perdería todo el dinero. Este modelo de consenso se basa en que los nodos que mayor garantía comprometen son los más interesados en que la blockchain funcione correctamente.

Proof of Stake, al requerir de menos recursos que PoW, tiene el potencial de ser mucho más escalable.

Uno de los problemas que surgen con el reparto de recompensas con este sistema es que, a diferencia de PoW donde, si un nodo tiene muchos recursos computacionales, siempre se le podría hacer competencia comprando más hardware, quitándole así cuota de minado y poder

en la blockchain, en PoS el que más moneda tiene siempre validará más bloques y recibirá mayores recompensas, no pudiéndosele arrebatar este poder si no las quiere vender.

Ataque de nothing-at-stake

Como se ha comentado, en una cadena que utiliza PoS, no se consumen muchos recursos para minar un bloque. Esto hace que, si un nodo malicioso quiere intentar hacer una bifurcación de la cadena, no tenga un coste (en términos energéticos) relevante. En PoW se requiere de minar un hash e intentar superar a la cadena más larga, si el intento es fallido, se pierde energía y tiempo de minado en la cadena principal, por lo que es un trabajo contraproducente. Sin embargo, en PoS, al no haber coste, una actitud racional sería intentar crear una bifurcación constantemente.

No obstante, existen propuestas que solucionan este problema, Ethereum ha propuesto una técnica llamada Slasher que permite el castigo a los mineros que intenten bifurcar la cadena. [25]

Algunas blockchains que utilizan Proof of Stake son Peercoin y Cardano. Ethereum implementará PoS próximamente, en su actualización a Ethereum 2.0.

1.2.4.3. Otros algoritmos de consenso

Para mejorar la escalabilidad de la blockchain, se han creado mecanismos por los cuales, limitando la descentralización, se consigue una mayor velocidad a la hora de generar y validar bloques nuevos, además de reducirse las comisiones que se deben pagar por el uso de la red.

La Binance Smart Chain (BSM) es una blockchain cuyo algoritmo de consenso se denomina *Proof of Stake Authority (PSA)*. En esta blockchain la generación de nuevos bloques es similar a PoS, pero con la diferencia de que el número de nodos validadores está limitado a 21. Para poder ser validador se debe poseer una dirección con una cantidad de monedas que esté en el Top 21. Además, los usuarios individuales pueden apoyar a los validadores que consideren confiables enviándoles sus propias monedas, ayudándole a conseguir el puesto y compartiéndose con ellos las recompensas.

La seguridad se sostiene sobre la reputación de los nodos, que además reciben un castigo si se detecta un comportamiento fraudulento.

1.3. Bitcoin. Blockchain de primera generación

Bitcoin es una forma de dinero electrónico puramente peer-to-peer que permite enviar pagos online directamente entre partes y sin pasar a través de una institución financiera. Para ello hace uso de las firmas digitales, pero además, para solucionar el problema del doble gasto sin recurrir a un tercero, la red sella las transacciones en el tiempo mediante el uso de una cadena de proof-of-work basada en hash, estableciendo un registro que no se puede modificar sin rehacer la proof-of-work. La cadena más larga sirve de prueba efectiva de la secuencia de eventos y demuestra que procede del conjunto de CPU más potente.

En resumen, la primera generación de blockchain consiste en la aplicación de esta tecnología en su versión más simple, un libro contable donde solamente se registran transferencias de monedas entre personas, de forma segura y confiable.

1.3.1. Características principales

1.3.1.1. Direcciones

Una dirección de Bitcoin es un identificador que contiene entre 27 y 34 caracteres alfanuméricos. Por ejemplo: 1A1zP1eP5QGefi2DMPTfTL5SLmv7DivfNa

Las direcciones se pueden crear de forma offline (de hecho, es el método más seguro, ya que se reduce el riesgo de que alguien pueda robar la información) ya que se generan realizando un proceso matemático mediante el cual se obtiene una clave pública y otra privada. La dirección aparecerá en la blockchain una vez se realice una transferencia a esta cuenta.

En Bitcoin las direcciones se crean de la siguiente manera:

1. Se genera una clave privada partiendo de una serie de números aleatorios
2. Se genera una clave pública utilizando el algoritmo de curva elíptica ECDSA.
3. Se realiza el hash SHA-256 de la clave pública.
4. Se realiza un nuevo hash sobre el resultado, en este caso con el algoritmo RIPEMD-160.
5. Se añade el byte de versión al inicio (0x00). Permite diferenciar las direcciones de Bitcoin de otras criptomonedas que utilizan el mismo sistema.

Ahora, al resultado obtenido se le requieren añadir 4 bytes más como “checksum”, para verificar que la dirección está bien escrita:

6. Se aplica el hash SHA-256 al resultado del punto 5.
7. Se aplica, de nuevo, un hash SHA-256 al resultado.
8. Se toman los 4 primeros bytes del resultado.
9. Ahora estos 4 bytes se añaden al final del resultado obtenido en el punto 5.
10. Finalmente, el resultado se transforma de bytes a caracteres alfanuméricos utilizando la codificación Bse58Check.

[26] [27]

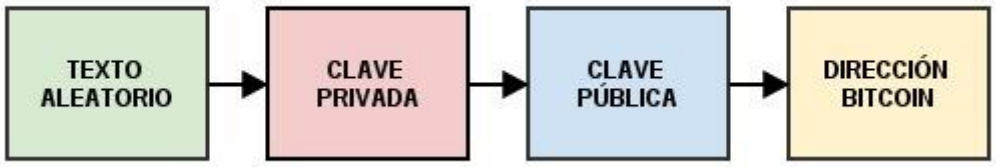

Figura 15. Generación de una dirección de Bitcoin. Fuente: Bit2me

1.3.1.2. Transacciones

El Bitcoin se puede dividir hasta su cienmillonésima parte, a esta unidad se la conoce como un Satoshi en honor a su creador (0,000000001 BTC = 1 Satoshi).

Las transacciones en bitcoin son algo más complejas que un intercambio de monedas de una dirección A a una dirección B.

Una wallet de bitcoin no contiene una sola dirección, sino muchas, que son generadas a partir de una misma raíz de números aleatorios.

Cada vez que se recibe un pago, por lo general se usa una dirección diferente (aunque se puede usar la misma) de la wallet, de forma que los bitcoins recibidos suelen quedar fragmentados entre todas estas direcciones.

A la hora de realizar un envío, se toman como inputs de la transacción un conjunto de direcciones de la wallet que contengan una cantidad superior o igual a la cantidad de bitcoin que se quiere enviar. Estas direcciones se vaciarán completamente, por lo que existirá un resto, la suma de todo el bitcoin recopilado menos la cantidad enviada al destinatario.

Como outputs, se toma la dirección del destinatario y otra dirección del conjunto que contenía nuestra wallet. A esta segunda, se enviará el resto de bitcoin no gastado. En el proceso también se debe pagar la comisión de la transacción al minero.

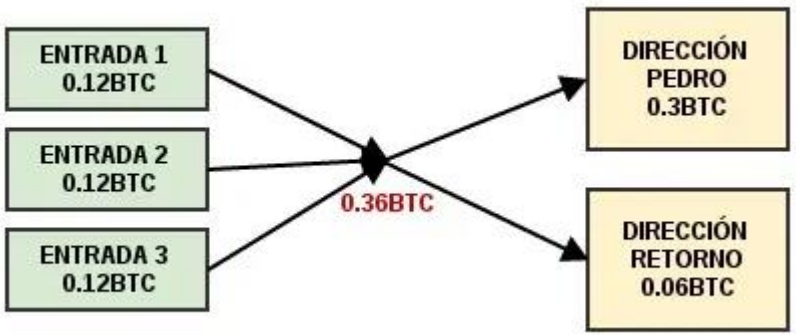

Figura 16. Transacción de Bitcoin. Fuente: Bit2me

Este proceso se realiza de forma automatizada por el programa que gestiona la wallet. Es una forma de garantizar la privacidad, ya que, al ser las transacciones públicas, de esta forma se dificulta el seguimiento de los fondos.

1.3.2. Las ventajas de Bitcoin

- Descentralización: Bitcoin no está controlado por ningún estado ni se requiere como intermediaria ninguna institución financiera. Esto significa que el propietario de la moneda puede utilizar sus fondos de forma libre y segura sin requerir de terceras personas.
- Inconfiscable, infalsificable y seguro: Nadie puede tener acceso o bloquear tus fondos, ni siquiera una entidad bancaria o un gobierno.
- Transacciones internacionales ágiles: Para realizar una transacción solamente se requiere de una conexión a internet y la transferencia queda registrada de manera casi instantánea. No es necesario utilizar el lento y costoso sistema bancario. [28]

La popularización en el uso de bitcoin ha hecho que la red tenga muchas solicitudes de transacción y, al ser los bloques de tamaño finito (1MB), la comisión para incluir una transacción sea cada vez mayor. A fecha de hoy, la comisión oscila entre los 15 y 25 euros, lo que hace inviable el uso de la blockchain para transferencias de pequeño valor.

21 millones de bitcoins

En el proceso de minado de un bloque, el minero además de las comisiones recibe bitcoins de nueva creación. Sin embargo, esta cantidad disminuye a la mitad cada cuatro años en un proceso que se conoce como *halving*, hasta llegar a ser 0 en el año 2144.

Actualmente el número de bitcoins en circulación es de 18.697.218, una vez se llegue a la fecha establecida se emitirá el último bitcoin, el número 21.000.000. Esto significa que el Bitcoin es un activo deflacionario

# HALVING	# BLOQUE	RECOMPENSA ACTIVA	# BTC INTRODUCIDO	AÑO
0	0	50	10500000,00000000	2008
1	210000	25	5250000,00000000	2012
2	420000	12,5	2625000,00000000	2016
3	630000	6,25	1312500,00000000	2020
4	840000	3,125	656250,00000000	2024
5	1050000	1,5625	328125,00000000	2028
10	2100000	0,04882813	10253,90625000	2048

15	3150000	0,00152588	320,43457031	2068
20	4200000	0,00004768	10,01358032	2088
25	5250000	0,00000149	0,31292439	2108
30	6300000	0,00000005	0,00977889	2128
34	7140000	0,00000000	0,00061118	2144
		TOTAL DE BITCOINS	**20.999.999,99938880**	

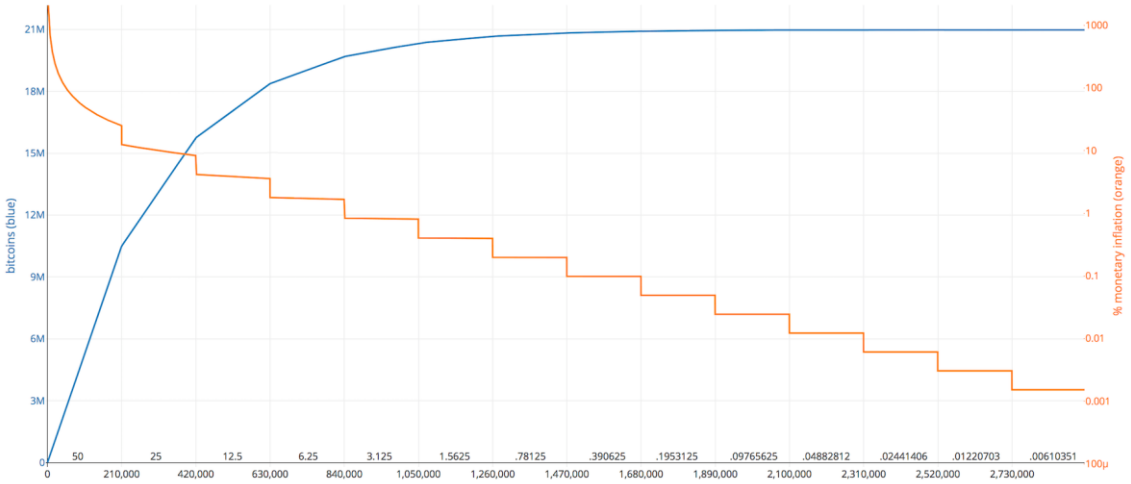

Figura 17. Número total de bitcoins a lo largo del tiempo.

1.4. Ethereum. Blockchain de segunda generación.

Pocos años después de la aparición de Bitcoin, los desarrolladores de blockchain se dieron cuenta de que utilizar esta tecnología exclusivamente para hacer pagos online era limitarse demasiado. Fue entonces cuando Vitalik Buterin, publicó el whitepaper de Ethereum, en el año 2013 a la edad de diecinueve años.

La principal característica de Ethereum es que, utilizando la tecnología blockchain, no sólo se pueden realizar transacciones de su moneda, el Ether, sino que también se pueden desplegar programas con los que los usuarios pueden interactuar. A estos programas se les denominan Smart contracts o contratos inteligentes. En resumen, Ethereum es una computadora descentralizada.

1.4.1. Smart contracts

Un contrato es un acuerdo legal, manifestado entre dos o más partes que se vinculan al mismo regulando así sus relaciones. Los contratos están sujetos a la ley y a la confianza entre los firmantes.

Un Smart contract, a diferencia de un contrato convencional, es capaz de ejecutarse y hacerse cumplir por sí mismo, de manera autónoma y automática, sin intermediarios ni mediadores, ya que simplemente se trata de códigos informáticos que solamente obedecen a la lógica contenida en ellos.

Con un Smart contract pueden interactuar personas, máquinas u otros Smart contracts que funcionen de forma autónoma. Su código es público y no se puede cambiar al estar sellado sobre la tecnología blockchain. Su naturaleza lo hace descentralizado, transparente e inmutable. [29]

De hecho, Ethereum es un sistema determinista y en él no puede existir la aleatoriedad ni se pueden programar variables aleatorias. Toda acción sobre la blockchain de Ethereum debe poder ser replicada por todos los participantes obteniéndose siempre el mismo resultado, de lo contrario no se podrían validar las transacciones al no haber consenso entre los nodos.

La Ethereum Virtual Machine (EVM)

La EVM es una máquina virtual que forma parte del ecosistema blockchain de Ethereum y es la que permite dar vida a los Smart contracts.

Para realizar la programación sobre la EVM, se desarrolló un lenguaje de programación llamado Solidity, muy similar a JavaScript.

1.4.2. Direcciones en Ethereum

En Ethereum existen dos tipos de cuentas: las cuentas de usuario y las cuentas de contrato. Cada una de las cuentas está identificada por una dirección que es única. Las cuentas de usuario únicamente pueden recibir Ether, mientras que las de contrato, además de eso, pueden ser requeridas para ejecutar funciones internas, tanto de lectura como de escritura. Estos requerimientos y envíos se conocen como transacciones y pueden hacerlos las cuentas de usuario y los propios contratos de forma automatizada. Cuando un contrato realiza transacciones con otro, se le denomina “transacción interna”.

La dirección pública de una cuenta de usuario es directamente la clave pública que surge de realizar el hash Keccak-256 a una clave privada (no es necesario realizar cálculos adicionales como en Bitcoin). La del contrato se determina a la hora de realizar su despliegue en la blockchain.

Un ejemplo de dirección en Ethereum es: 0x60A2823E92c2d3d8E9B641e433B6b53b897244D7

1.4.3. Economía de la blockchain de Ethereum

Ejercer como nodo validador en la blockchain supone un coste computacional. Este coste proviene principalmente del esfuerzo para obtener un hash válido mediante PoW y de la ejecución de las instrucciones de los Smart contracts. El procesamiento necesario para interactuar con un Smart contract depende de las características propias del contrato, por lo

que las operaciones más complejas serán más caras. En Ethereum, la unidad en la que se contabiliza este esfuerzo computacional es el Gas y éste tiene un precio que se valora en Ether, la moneda propia de la blockchain. El minero que valide el bloque recibirá como pago 2 Ethers de nueva creación y la comisión de la transacción, que será $Gas\ consumido \cdot Precio\ del\ Gas$

El precio del gas, al igual que las comisiones en Bitcoin, depende de lo saturada que esté la red, ya que los mineros tomarán como prioritarias las transacciones que paguen más, a la hora de incluirlas en la blockchain.

El Ether se puede subdividir en 18 cifras decimales, a esta unidad más pequeña se le denomina Wei y normalmente las operaciones en Ethereum se expresan en Gigaweis (GWei).

$$1\ Ether = 1.000.000.000\ Gwei = 1.000.000.000.000.000.000\ Wei$$
$$0{,}000000001\ Ether = 1\ Gwei = 1.000.000.000\ Wei$$

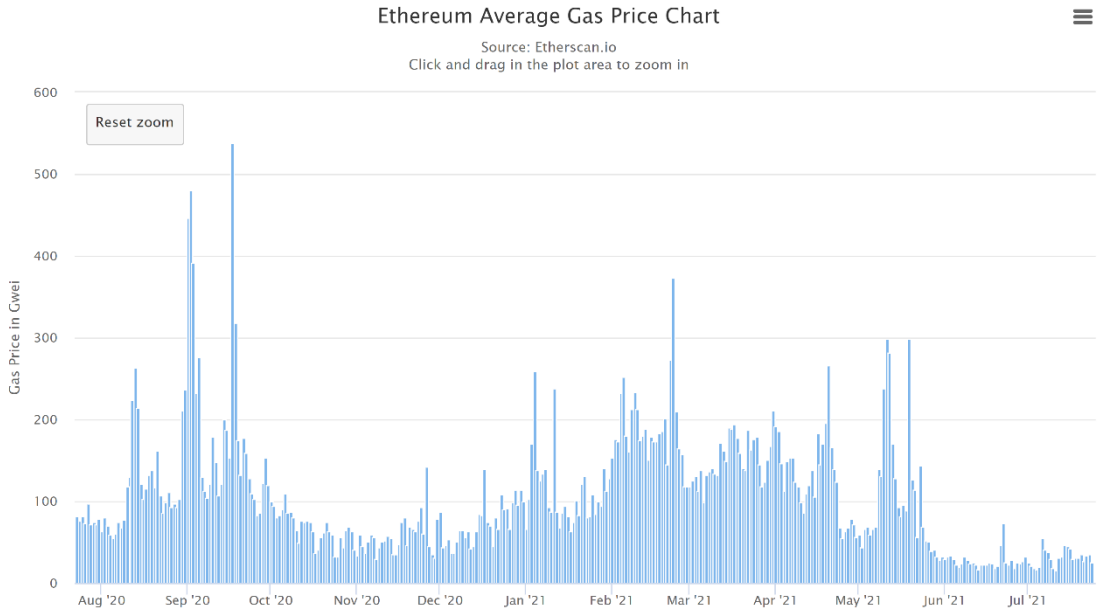

Figura 18. Precio del gas a lo largo del año 2020 - 2021. Fuente: Etherscan.io

1.4.4. Ventajas de Ethereum

- Blockchain multipropósito gracias a la capacidad para integrar y utilizar Smart contracts.
- Los bloques se minan cada 15 segundos, por lo que tiene una alta velocidad a la hora de confirmar transacciones.
- Su desarrollo se controla de forma descentralizada y no existe ninguna autoridad central que regule su funcionamiento.
- Es una blockchain pública y cualquiera puede convertirse en un nodo validador, realizar transacciones, escribir contratos o interaccionar con ellos si así lo desea.

1.4.5. Problemas de Ethereum

El principal problema que sufre Ethereum es la saturación de la cadena de bloques. La red solamente puede manejar unas 15 transacciones por segundo, lo que sumado a su alta popularidad da como resultado precios del Gas, y en consecuencia comisiones, extremadamente altos. Actualmente la realización de una transacción simple equivale a lo que serían 5 dólares, la interacción con un Smart contract de 10 a 30 dólares (depende de la función) y el despliegue de un contrato no menos de 100 dólares.

Los desarrolladores siguen trabajando en Ethereum para mejorar la escalabilidad de la cadena de bloques y que su acceso sea más económico. Se esperan una serie de actualizaciones a lo largo de 2021 y 2022 para evolucionarla a la llamada Ethereum 2.0 donde la mejora más relevante es la transición de PoW a PoS. Se espera que las transacciones por segundo puedan llegar hasta las 100.000.

Poseer un nodo completo de Ethereum requiere de 5TB de almacenamiento (es la cadena de bloques que más ocupa) y una conexión de 100Mbps, por lo que, para interactuar con ella, la mayoría de los usuarios recurre a nodos externos especializados a los que envía las transacciones firmadas para que sean incluidas en la cadena, lo que podría perjudicar a la descentralización. [30]

1.5. Blockchain de tercera generación

Desde la creación de Ethereum han ido surgiendo muchos proyectos de blockchain que intentan solucionar los problemas que ésta experimenta.

Los aspectos clave a mejorar son la escalabilidad, la interoperabilidad (comunicación entre blockchains diferentes), la sostenibilidad, la velocidad en las transacciones, los costos de envío y la eficiencia de la red.

Los proyectos que mayor confianza reciben por parte de la comunidad de usuarios e inversores son:

- *Cardano:* Fue creado por el matemático Charles Hoskinson, cofundador de Ethereum. Cardano una plataforma que utiliza PoS y pretende alcanzar cientos de miles de transacciones por segundo.
- *Polkadot*: También fundada por un cofundador de Ethereum, Gavin Wood, inventor del lenguaje Solidity. También pretende alcanzar un número de transacciones similar al de Cardano.

1.6. Aplicaciones de la blockchain

1.6.1. Reserva de valor

Bitcoin es el primer activo monetario real, intangible, descentralizado y privado de la historia de la humanidad.

- Activo monetario: Puede utilizarse para reducir los costes de transacción de los intercambios comerciales
- Real: Es un activo real porque no es el pasivo de ningún otro agente. Esto significa que no es la deuda de nadie, al igual que el oro, inmuebles, etc. Lo opuesto serían activos financieros como los bonos, las acciones o el dinero Fiat.
- Intangible: No depende del soporte material en el que se halle corporeizado, sino de la realidad subyacente que representa. Otros activos intangibles serían patentes o marcas.
- Descentralizado: Tanto su emisión es descentralizada, como los intercambios, que no dependen de un tercero.
- Privado: Es privado en el sentido de que ha surgido de forma espontánea en el mercado y no ha sido creado ni se ha incentivado su uso por parte de ningún Estado, como si pudiera serlo el Yuan digital, la criptomoneda que ha lanzado el gobierno chino.

Estas características, únicas en su conjunto, junto con otras cualidades deseables en “el dinero”, como el ser deflacionaria (máximo de 21 millones), la gran divisibilidad (en 8 cifras decimales) y bajos costes de transferencia y almacenamiento, hacen que el Bitcoin empiece a ser considerado como un lugar seguro en el guardar los ahorros.

Para que esto acabe convirtiéndose en una realidad consistente a largo plazo, es necesario que la confianza en la moneda y la tecnología que la respalda se generalice formando un “efecto red”. [31]

1.6.2. Stablecoins

La extrema volatilidad de las criptomonedas puede generar rechazo entre muchos potenciales usuarios, a los que les gustaría hacer uso de las virtudes de una moneda digital sin verse expuestos al riesgo que suponen cambios de precio muy agresivos.

Para resolver el problema de la volatilidad surgieron las stablecoins o monedas estables. Son criptomonedas respaldadas por algún mecanismo que hace que su precio se mantenga estable, generalmente siguiendo el valor del dólar estadounidense.

1.6.2.1. Monedas respaldadas por moneda fiat

Este tipo de criptomoneda está respaldada 1:1 con una moneda real. Una empresa u organización se encarga de controlar la emisión o quema del activo digital en función de su demanda. Un ejemplo sería Tether (USDT), respaldado por dólares.

Este mismo mecanismo de respaldo se puede realizar con acciones, materias primas, etc. De esta forma se puede transferir cualquier activo al mundo digital y representarlo en una moneda que siga su valor. Digix Gold (DGX) lo hace con el oro, 1DGX = 1 gramo de oro.

Un problema de este tipo de monedas es que se debe confiar en el buen hacer de la organización que custodia el activo que representan.

1.6.2.2. Monedas respaldadas por criptomonedas

Para evitar tener que confiar en una organización externa, se ideó un sistema para mantener estables los precios sin la necesidad de una intermediación ajena.

Este tipo de criptomonedas están respaldadas por otras criptomonedas. Para su generación es necesario depositar en un contrato inteligente otra criptomoneda, de precio variable.

Un ejemplo es MakerDAO, una plataforma que, mediante un contrato inteligente (totalmente automatizado y público), emite la moneda DAI, que equivale a 1 USD.

Supongamos que entregamos al contrato 1Ether = 2000$. El contrato nos devolvería 1500 DAI, una cantidad algo inferior al valor real, para tener un margen de seguridad frente a variaciones en el precio del Ether.

Si el valor de mercado del Ether sube a 2500$, no pasaría nada, podríamos incluso demandar más DAI. Sin embargo, si el valor desciende a valores cercanos a 1500$, se produce una “margin call”. Se pide al usuario que aporte más Ether para mantener la operación activa, si no lo hace, el Ether se vende en el mercado a cambio de los 1500 DAI y se cierra la operación. Cabe destacar que independientemente de la persona que haya generado el DAI, éste puede intercambiarse en la blockchain libremente con la confianza de que su valor siempre será de 1 dólar.

Ahora la confianza no se deposita en una organización, sino en la buena programación del Smart contract que gestiona las operaciones, que como es público, todo el mundo puede verificar. También se debe confiar en el buen seguimiento de precios de mercado de las criptomonedas que se usan como respaldo, información que se obtiene del mundo real mediante el uso de un Oráculo descentralizado, del que se hablará más adelante.

1.6.3. Utility tokens. Tokens de utilidad

Es un tipo especial de token que sirve de ayuda en la capitalización o financiación de proyectos para startups. Este tipo de activo se emite con la celebración de una ICO (Initial Coin Offering – Oferta Inicial de Monedas) donde los inversores pueden comprarlos, de forma similar a como una empresa emite nuevas acciones.

El valor intrínseco del token viene dado por la actividad de la empresa que lo respalda. Generalmente el token es necesario para poder tener acceso a los servicios o productos de la empresa. En otros modelos, la empresa utiliza sus beneficios para comprar esos tokens en el mercado y destruirlos, reduciendo su número en circulación, ésta es una forma de repartir dividendos, similar a la recompra de acciones de las empresas.

La moneda, en ocasiones, también permite a los poseedores votar en decisiones que afectan al funcionamiento de la empresa. [32]

Algunos ejemplos son Basic Attention Token (BAT), que sirve como medio de pago a los creadores de contenido y usuarios por poner publicidad en internet por parte de empresas, en el navegador Brave.

Filecoin (FIL) sirve como medio de pago para servicios de almacenamiento descentralizado en internet.

Otros como la plataforma Aave, que se dedica, entre otras cosas, dar servicios de préstamo, utiliza los beneficios obtenidos de las comisiones en comprar su token AAVE en el mercado. Los propietarios del token también pueden votar en los procesos de gobernanza. [33]

1.6.4. Security tokens. Tokens de propiedad

Los security tokens, son un tipo de tokens criptográficos vinculados a los security tradicionales. Están relacionados con valores financieros como bonos, swaps o futuros. A diferencia de los "utility tokens" no son un medio, ni el resultado de la actividad de una empresa, sino que son la representación criptográfica de contratos de inversión y tienen validez jurídica. Los gobiernos están trabajando en su regulación para que sigan normas similares a cualquier activo financiero. [34]

El beneficio de usar security tokens está en la posibilidad de llegar a un público más amplio, eliminar la burocracia y la lentitud. Todo a un coste más reducido que tener que cotizar acciones. [35]

1.6.5. Tokens en Ethereum

Cada una de estas criptomonedas o tokens, pueden pertenecer a una blockchain propia (como Filecoin), en este caso se denomina "criptomoneda", o, por medio de un contrato inteligente, crearse dentro de la blockchain de Ethereum (u otra blockchain que permita contratos inteligentes), en este caso se denomina "token".

En Ethereum, la comunidad de desarrolladores ha establecido estándares para la creación de diferentes tipos de tokens. De esta forma, al crear distintos tokens con un mismo estándar, se facilita la programación y la implementación de funcionalidades que interactúen con éstos, facilitando el trabajo de los desarrolladores.

Los estándares más habitualmente utilizados son ERC-20 y ERC-721 y los tokens enunciados anteriormente suelen ser adaptados a una de estas dos estructuras para ser incorporados a la blockchain de Ethereum.

1.6.5.1. Tokens fungibles. ERC-20

Los tokens fungibles se caracterizan porque pueden fraccionarse y cada unidad es igual a las demás. Esto significa que pueden mezclarse, agruparse e intercambiarse sin distinción. Ejemplos de tokens fungibles en el mundo real serían las divisas, las acciones, los bonos, etc. En la blockchain de Ethereum, siguen el estándar ERC-20 monedas como DAI, AAVE, etc.

1.6.5.2. Tokens no fungibles (NFT). ERC-721

Los tokens no fungibles se caracterizan porque cada uno de ellos tiene un identificador que lo hace único. Pueden intercambiarse de forma individual y no pueden mezclarse.

Inicialmente su propósito fue el de vincularse a elementos coleccionables en el mundo digital, sin embargo, ahora se están empezando a utilizar para representar objetos del mundo real y vincularlos a un registro o una serie de datos. También pueden exhibir unas características o un comportamiento propios, gestionados por el Smart contract que los creó.

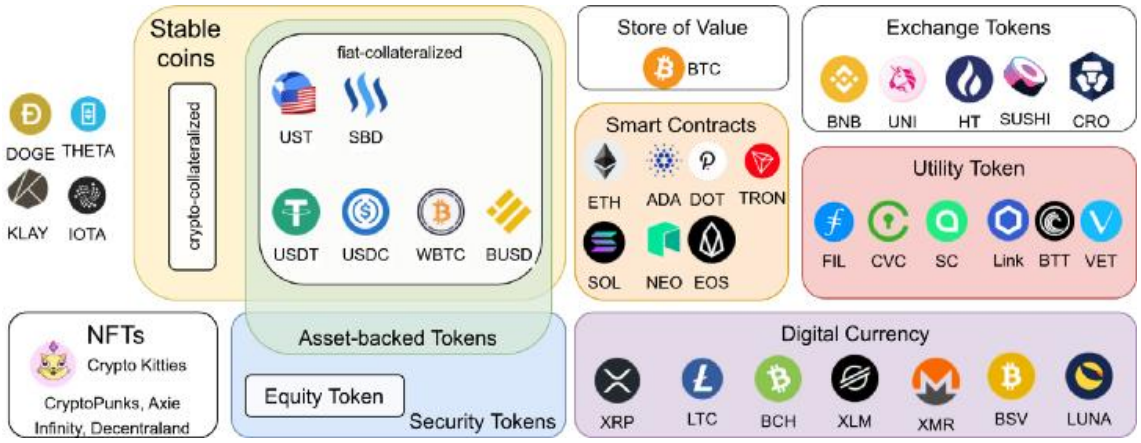

Figura 19. Algunos tipos de monedas y tokens en la blockchain. Por Martin Thoma. [36]

1.6.6. Oráculos descentralizados

Un oráculo es un servicio ofrecido por un tercero que sirve para aportar información externa a un Smart contract dentro de la blockchain. Es una manera de unir al mundo exterior con la cadena de bloques. Sin los oráculos, los Smart contracts sólo tendrían acceso a información interna ya almacenada en la blockchain y perderían gran parte de su utilidad.

Un oráculo no es la fuente de información en sí misma, sino que consulta fuentes externas, las verifica y las introduce en un Smart contract destinado a recibir los datos, realizando una transacción.

Al utilizar un oráculo estamos interactuando con una tercera persona, que nos debe proveer de información veraz, lo cual puede generar problemas de confianza. Cuando este proceso de

consulta y verificación se produce mediante mecanismos descentralizados, hablamos de un oráculo descentralizado. Este modelo permite mantener lo máximo posible la esencia descentralizada de la blockchain.

Un oráculo descentralizado funciona de la siguiente manera. Todo el proceso es mediado por una serie de Smart contracts.

- Se solicitan una serie de datos externos (por medio de un Smart contract) y se envía un token propio del oráculo como pago.
- El oráculo solicita esta información a un conjunto de nodos que participan en su blockchain propia.
- Los nodos consultan la información y envían una respuesta.
- El oráculo toma la mediana de todas las respuestas y reparte el pago entre los nodos, premiando a los que más se hayan acercado a la mediana. Si algún nodo emite una respuesta atípica se le castiga con una multa sobre los fondos que haya puesto como garantía para poder acceder a su puesto de nodo.
- La información es entregada al Smart contract que la solicitó.

El oráculo más popular que trabaja en la blockchain de Ethereum es Chainlink y su token propio es LINK. [37]

1.7. Riesgos del ecosistema blockchain

La tecnología blockchain puede ayudar a mejorar los procesos productivos, la logística y revolucionar el sistema bancario. Sin embargo, como toda tecnología emergente, lleva aparejados unos riesgos que deben ir mitigándose a medida que avanza su evolución. Los problemas que pueden surgir son similares a los que se enfrentaba Internet a finales del siglo pasado.

1.7.1. Complejidad tecnológica

Blockchain es una tecnología nueva y compleja. Empezar a realizar transacciones en la blockchain no es especialmente difícil, pero requiere de un proceso de aprendizaje que, de momento, muy poca gente ha iniciado. Tampoco hay muchos programadores, en relación con otras tecnologías, aunque tampoco había casi desarrolladores de aplicaciones para smartphone hace diez años. A medida que pase el tiempo, más gente participará en la implementación y uso de la tecnología, quienes trabajarán para que el acceso a la tecnología sea lo más sencilla posible, creando aplicaciones más intuitivas y de fácil acceso.

Sin embargo, no se puede obviar el hecho de que es posible que haya personas que nunca adopten su uso o tarden mucho más tiempo del esperado en hacerlo. Incluso que las expectativas puestas hoy en día no lleguen a cumplirse.

1.7.2. La descentralización

Hasta ahora se ha hablado de la descentralización como una virtud de la blockchain, pero la descentralización lleva aparejados riesgos. El hecho de no depender de un tercero a la hora de realizar transacciones o ejecutar un contrato, hace que no se requiera la confianza de nadie ajeno a ti mismo para realizar las operaciones que desees, pero también te hace plenamente responsable de lo que estás haciendo. Es el usuario el responsable de mantener seguras sus claves privadas, ya que, si alguien las roba, no hay vuelta atrás, nadie podrá intermediar y recuperar los fondos. Si se escribe un contrato, es el usuario el responsable de que no existan vulnerabilidades y que funcione como debe hacerlo, ya que nadie podrá interceder y reinterpretarlo.

Este riesgo se ve potenciado por el anterior, ya que, al ser una tecnología joven y compleja, se pueden cometer errores más fácilmente.

1.7.3. Alta volatilidad

Las criptomonedas y los tokens son activos nuevos en el mercado y por ello están sometidos a mucha especulación. Pueden experimentar subidas y bajadas en el precio muy agresivas. Las stablecoins y el incipiente mercado bancario sostenido en la blockchain tratan de reducir estos riesgos. No obstante, para realizar, por ejemplo, operaciones en Ethereum, se necesita depender de la compra de Ether. Se puede entender como una materia prima necesaria para establecer un modelo de negocio en la blockchain y, por tanto, variaciones inesperadas en su precio pueden ser un riesgo importante a tener en cuenta.

1.7.4. Privacidad

Una blockchain pública, a la que tiene acceso libre cualquier usuario, también contiene información pública. Esto implica que es posible realizar un seguimiento del movimiento de fondos a través de distintas direcciones. Las direcciones de la blockchain no están “a nombre” de nadie, pero en el caso de que se relacione una dirección con una persona concreta, se podrían conocer sus posesiones.

La información contenida en un Smart contract, con mayor o menor dificultad también se puede conocer.

Se está trabajando en mecanismos para hacer más difícil el seguimiento de fondos. Por ejemplo, Monero es una criptomoneda en cuya blockchain, al realizar una transacción, esta se “mezcla” con las transacciones de otros usuarios, haciendo que se rompa el seguimiento.

La información de un Smart contract puede codificarse antes de enviarse para evitar que alguien ajeno “entienda” su contenido.

1.7.5. Fraude

La blockchain, al ser una tecnología nueva, compleja y de libre acceso por cualquier ciudadano, puede generar un entorno donde estafadores intenten engañar a los usuarios para apropiarse de sus fondos.

Algunas formas de fraude incluyen el embaucar a algún usuario para que envíe su dinero a una cuenta antes de recibir el pago pactado. Una vez realizado el envío no hay forma de revertirlo y es casi imposible identificar al estafador.

Otras consisten en interacciones con Smart contracts fraudulentos, cuya lógica funciona de forma diferente a la prometida. Si la empresa que lo generó no es de confianza o no se conoce su programación interna, estaríamos enfrentándonos al riesgo de estar dando acceso a nuestros fondos sin saberlo.

1.7.6. Regulación

Los gobiernos de algunos países podrían sentirse amenazados por la existencia de potenciales sustitutos de sus monedas nacionales sobre los que no tienen control, o considerar que ciertos actores pueden hacer un uso delictivo de la blockchain. Pese a no ser posible por su parte bloquear el acceso a las blockchains públicas, pueden mostrarse hostiles y dificultar el acceso a ellas.

Un ejemplo es la prohibición del minado de Bitcoin y la oferta de servicios financieros relacionados con criptomonedas por parte de China [38]. Aunque, por otra parte, el gobierno chino sí parece interesado en otras aplicaciones de la tecnología blockchain, como la que ofrece Vechain. [39]

Por ello, urge que los gobiernos ofrezcan seguridad jurídica y comuniquen de forma abierta su visión e intenciones respecto a esta nueva tecnología.

1.8. Modelos de negocio basados en blockchain

En términos generales, en el mundo del blockchain se pueden diferenciar cuatro aplicaciones fundamentales sobre las que construir un modelo de negocio. Todas ellas pueden interactuar entre sí y en cierto sentido están relacionadas.

1.8.1. Finanzas descentralizadas

Es lo que se conoce popularmente como DEFI (Decentralized Finance). Aprovechando las características de transparencia y descentralización de la blockchain, se pueden ofrecer todo tipo de servicios financieros.

Los más populares son los de intercambio de divisas (en el caso de la blockchain, intercambios entre sus tokens), lo que ayuda a manejar el valor de éstos y facilita su compraventa. También son muy utilizados los servicios de préstamo, en los que dejando como colateral un token, se puede pedir prestado otro. Evidentemente quién presta cobra un interés pagado por el prestatario. Actualmente es fácil conseguir rendimientos anuales del 10% en monedas estables.

Activos	Tamaño de mercado	Total prestado	Interés de depósito (APY)	Interés de préstamo (APY) Variable
DAI	1,69B	1,36B	2,99 % 1,23 % APR	4,10 % 1,55 % APR
Gemini Dollar (GUSD)	33,33M	26,56M	2,86 % 3,41 % APR	3,98 % 4,28 % APR
USD Coin (USDC)	5,71B	5,18B	6,66 % 1,14 % APR	8,13 % 1,27 % APR
USDT Coin (USDT)	1,08B	983,54M	10,10 % 3,74 % APR	12,30 % 4,14 % APR
Ethereum (ETH)	1,9M	86,54K	0,02 % 0,50 % APR	0,56 % 0,59 % APR
WBTC Coin (WBTC)	42,18K	2,34K	0,03 % 1,19 % APR	0,68 % 1,14 % APR

Figura 20. Tipos de interés pagados para diferentes criptomonedas. Entre ellas, monedas estables vinculadas al dólar. Fuente: Aave

Fuera de las finanzas, existen otros servicios que también tienen que ver con el "movimiento de dinero" como modelos basados en casinos online, loterías…

1.8.2. Propiedad

La tecnología blockchain es una gran herramienta a la hora de garantizar la propiedad de un bien. Cualquier activo del mundo puede ser "tokenizado", esto significa que en la blockchain se guarda un token que representa de forma única a este bien del mundo real. Se podrían tokenizar productos de consumo: como relojes, vehículos, maquinaria, alimentos…; viviendas, terrenos u obras de arte, entre otros.

La forma más común de hacerlo es poseyendo un token no fungible (NFT), al que se le incorpora la información de interés que pueda estar asociada al producto que representa.

Actualmente esta forma de tokenización se ha vuelto muy popular en el mundo del arte contemporáneo, donde el bien que se tokeniza es una obra de arte, tanto digital como real, para así garantizar que la persona que lo posee es el propietario. En este caso se juega con la realidad de que la obra como producto tangible no tiene valor, sino que éste reside en algo más abstracto. El valor está en lo que la obra representa y en la capacidad de demostrar que aquello que representa, te pertenece.

También es utilizado para garantizar la autenticidad y propiedad de productos de lujo, siguiendo la línea de que el valor no está en el producto, sino en lo que éste representa.

Se puede ver como este modelo de propiedad tiene carencias, ya que no se basa en el valor del producto en sí mismo, sino en el valor de ser propietario de éste. Esto se debe a que actualmente no existe una regulación que reconozca al propietario de un token como propietario de un bien físico de la misma forma que se haría tras un proceso administrativo. En algunos países ya se ha empezado a dar validez legal a la tokenización de ciertos productos financieros o bienes raíces, sin embargo, aún queda un largo recorrido.

Lo que es seguro, es que finalmente llegará el día en el que se generalice su uso, debido a su enorme utilidad. En el momento en el que la propiedad de un token que represente un activo real tenga validez legal, se podrá tokenizar prácticamente cualquier objeto y comerciar con él de forma rápida, sencilla y segura. No se necesitarán largos procesos administrativos para realizar un sencillo cambio de propietario, datos que, además, quedarán registrados para siempre en la blockchain de forma pública (o pseudopública), siendo de fácil acceso para cualquier agente y sin riesgos de manipulación o falsificación futura.

Todos los bienes tokenizados también se podrán introducir en el marco de las finanzas descentralizadas y se podría, por ejemplo, pedir una hipoteca dejando como colateral el token de una vivienda o un terreno, financiar la compra de un producto a plazos o alquilar un vehículo. Todos estos procesos de una forma transparente y completamente determinada por el contrato que los medie.

1.8.3. Bases de datos

Las bases de datos sirven para almacenar datos pertenecientes a un contexto determinado. La blockchain, como base de datos descentralizada que es, puede utilizarse directamente para este propósito. El valor diferencial por el que guardar la información en una, consiste en la capacidad de la blockchain para garantizar que la información es inmutable y de fácil acceso por todos los participantes.

Las aplicaciones más prometedoras se encuentran en el mundo de la logística. La tecnología blockchain puede utilizar para hacer un seguimiento en la cadena de suministro, donde se registre el origen de las materias primas, los componentes y todos los intermediarios a través de los cuales han pasado los productos. De esta forma se podrían automatizar envíos, pagos, desencadenar incidencias… a la vez que se le ofrecen al consumidor una serie de garantías por el producto que compra. Ya existen proyectos piloto en la industria alimentaria, vinícola o automovilística.

Otra aplicación está en la sanidad. Aquí hay propuestas para guardar el historial clínico de un paciente, al que se podría acceder en cualquier momento y lugar. También para controlar el consumo de medicamentos de forma automática en farmacias.

Se debe recordar que, si la información se guarda en una blockchain pública, cualquiera puede ver los datos guardados en ella. Si no hay funciones de lectura o no se conocen, puede ser complicado leerlas, pero alguien con los conocimientos suficientes podría descodificarla. En estas situaciones, podría utilizarse la función hash para codificar información estructurada o guardada en otra base de datos privada, de forma que un tercero podría confirmar si la información que le es entregada desde esa base de datos privada no ha sido modificada.

Guardar información en una blockchain pública puede resultar en un alto costo de gas, por lo que es recomendable reducir al máximo los bytes guardados para ahorrar costes.

1.8.4. Servicios descentralizados

Es posible descentralizar actividades empresariales en las que el contrato inteligente es el mediador y punto de conexión entre diferentes agentes. Las relaciones entre los participantes en el servicio, por ejemplo, clientes, proveedores, inversores… están definidas por la forma en la que el contrato inteligente está diseñado. En función de sus intereses concretos, cada agente tomará un rol habilitado por el contrato y obtendrá un pago o un beneficio según su actividad.

Todo esto, en otra situación sería gestionado por una empresa que actuaría como intermediaria.

Los proyectos más prometedores se basan en servicios de internet, aunque no necesariamente deben serlo.

Actualmente la mayor parte de los servicios de internet están centralizados por grandes compañías como Amazon, Google y Facebook con las que difícilmente se puede competir. Debido a esta centralización ostentan un gran poder, tanto económico (dominio del mercado), como social (tienen la capacidad de censurar y bloquear a usuarios y empresas en el acceso a sus servicios).

Sin embargo, utilizando la blockchain como soporte para establecer relaciones entre usuarios, están comenzando a surgir empresas que se dedican a crear un entorno descentralizado para la oferta de diferentes servicios.

Estos servicios vendrían desde la oferta de almacenamiento en la nube (Filecoin), infraestructura para vídeo en streaming (Theta), poder de computación (Golem), etc.

En el caso de Theta, los agentes serían publicistas, streamers, espectadores y mineros. Los publicistas realizarían un pago por mostrar publicidad, y al resto de los agentes les sería repartido ese pago en función de su aportación al sostenimiento de la blockchain de Theta, de la forma en la que se haya determinado previamente.

2. El servicio de post venta de un vehículo basado en blockchain.

2.1. Token de propiedad de un vehículo

Se va a crear un *token de propiedad* para identificar vehículos de forma fiable, segura y accesible en la blockchain.

El token en la vida real consiste en un conjunto de información o datos, que se identifican con un número único, y este número se asocia a una persona (o dirección en la blockchain). Para visualizarlo de una forma más intuitiva, al número y la información que contiene la denominamos *token* y decimos que este token lo posee la persona a la está asociado.

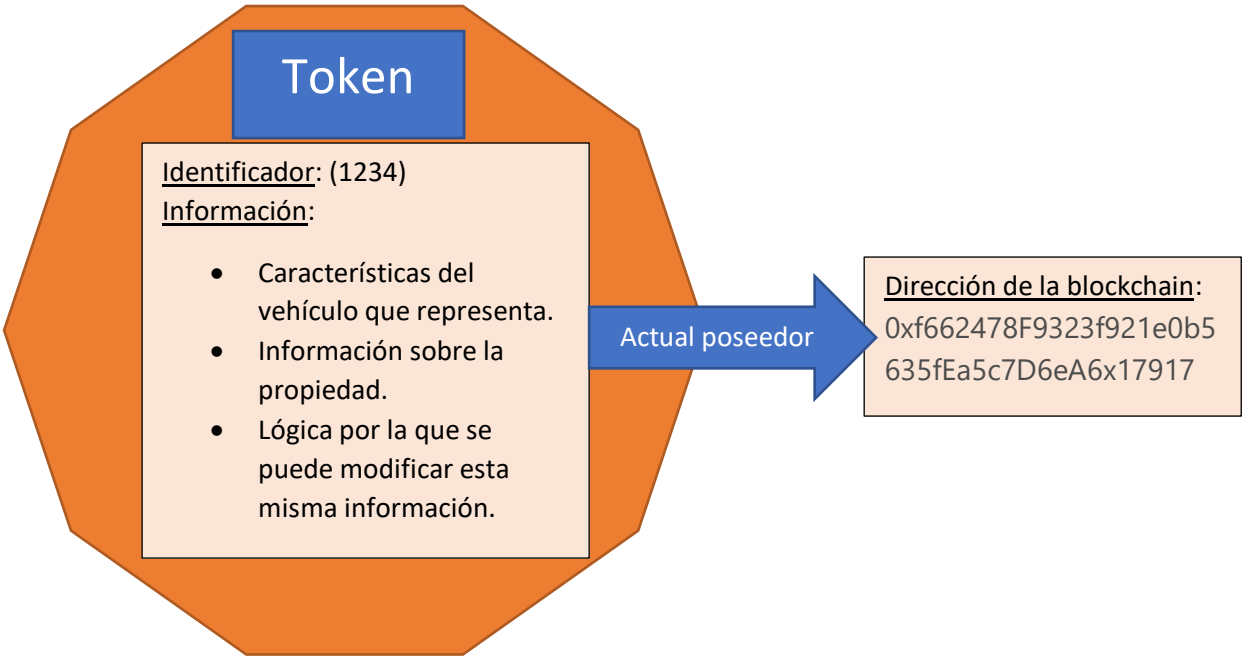

Figura 21. Esquema de un token. Fuente: Elaboración propia.

De esta forma, cuando se dice que un token se "vende" o "intercambia", en realidad solo se está cambiando la dirección a la que se asocia. Cuando realizamos "acciones" sobre el token, lo que se está haciendo es editar su información.

El identificador podría no incluirse si los tokens fuesen iguales y no importase tener uno u otro. Pero en nuestro caso, como cada token va a representar un objeto de la vida real (un vehículo) con características que van a ser específicas de cada uno de los vehículos, deben poderse diferenciar y manejarse independientemente. Si un token tiene este tipo de identificadores únicos, lo llamamos *token no fungible* o abreviadamente *token NFT*.

Para poder crear un token, es necesario generarlos con un *Smart contract*. El Smart contract no sólo permite crearlos, sino que puede agregar funcionalidades para establecer una lógica en la

forma en la que se cambia de propietario o en la que se edita y añade información nueva. Esta lógica se lleva a cabo mediante *funciones*.

En este trabajo, se va a utilizar esta idea de gestión de tokens e información para gestionar la información y la propiedad de vehículos. Cada token va a ser la representación en la blockchain de un vehículo y por medio de un Smart contract se va a ofrecer un servicio de postventa. El contrato inteligente además de emitir un token por cada vehículo tomará el papel de mediador en las interacciones entre usuarios y de los usuarios con los vehículos.

Figura 22. Esquema de un smart contract. Fuente: Elaboración propia.

Pueden existir diferentes mecanismos de programación para generar los tokens. Sin embargo, a lo largo del tiempo, los desarrolladores de este tipo de tecnologías han establecido unos estándares para ayudarse a la hora de programar e interactuar con los tokens de otros creadores. De esta manera, sin necesidad de conocer en detalle la programación interna de un token, pero sabiendo que contiene unas funciones preestablecidas, otros desarrolladores pueden crear aplicaciones que los manejen, o que un token recién creado sea compatible con ellas.

En nuestro caso, el estándar de programación que se utiliza para crear tokens con identificadores únicos, ósea, tokens NFT es el *estándar ERC-721*.

Este estándar cuenta con funciones ya establecidas para generar tokens nuevos, destruirlos, intercambiarlos…

Al tomar un estándar ampliamente probado, también reducimos el riesgo de que existan vulnerabilidades o errores de programación importantes, que, en el caso de existir, echarían abajo su usabilidad.

2.1.1. La lógica detrás de un token no fungible y el estándar ERC-721

A continuación, se va a describir el funcionamiento interno de un token no fungible. La programación está realizada con *solidity*, el lenguaje de programación que se utiliza en la blockchain de Ethereum.

La forma en la que se relacionan dos variables diferentes en Solidity es haciendo uso de un *mapping*. De esta forma, es posible asignar al identificador de cada token, la dirección de su legítimo propietario: Identificador => Dirección

```
// Mapping from token ID to owner address
mapping (uint256 => address) private _owners;
```

También se utilizan otros mappings para que quede registrado cuántos tokens tiene un usuario y a que terceros usuarios se les ha dado permiso para gestionar un token que no es suyo.

La propiedad y los permisos se pueden otorgar y retirar por medio de una serie de funciones bien definidas y que tienen en cuenta todos los supuestos.

2.1.2. Las funciones (más importantes) del estándar ERC-721

"constructor()", establece el nombre del token

Esta función solo se ejecuta una vez, en el momento de enviar el Smart Contract para ser guardado en la blockchain. Se utiliza para establecer variables de estado del contrato que no se volverán a modificar, como permisos de administrador o, en el caso de los tokens ERC-721 que genere este contrato, su nombre general y símbolo.

Si el constructor tiene variables de entrada, deben aportarse en el momento de su envío a la blockchain.

```
constructor (string memory name_, string memory symbol_) {
    _name = name_;
    _symbol = symbol_;
}
```

El símbolo y nombre que se definan en este momento identificarán a los tokens generados por este contrato y los diferenciará de los generados por otros, de una forma rápida y visual.

"mint()", la función generadora de tokens

Esta función se utiliza para generar un nuevo token. A la función se le aporta un número identificativo único (la función se asegurará primero de que no existe otro igual) y una dirección

a la que enviarlo inicialmente. Esta relación se incluirá en el mapping *Identificador => Dirección del propietario* que se ha visto antes.

A esta función no debe poder acceder todo el mundo, se restringirá su acceso a sólo las personas que el creador del Smart contract decida.

```
    function _mint(address to, uint256 tokenId) internal virtual {
        require(to != address(0), "ERC721: mint to the zero address");
        require(!_exists(tokenId), "ERC721: token already minted");

        _beforeTokenTransfer(address(0), to, tokenId);

        _balances[to] += 1;
        _owners[tokenId] = to;

        emit Transfer(address(0), to, tokenId);
    }
```

"burn()", la función destructora de tokens

Se utiliza para "quemar" o destruir tokens. Es posible que, llegado el momento, se quiera eliminar un token existente. Para este fin se utilizaría esta función de quema. Al igual que la función "mint" es necesario definir quién va a tener permiso para utilizarla.

```
    function _burn(uint256 tokenId) internal virtual {
        address owner = NFTVehiculo.ownerOf(tokenId);

        _beforeTokenTransfer(owner, address(0), tokenId);

        // Clear approvals
        _approve(address(0), tokenId);

        _balances[owner] -= 1;
        delete _owners[tokenId];

        emit Transfer(owner, address(0), tokenId);
    }
```

Transfiriendo la propiedad del token con “transfer()”

Es la función más importante del estándar ERC-721. Permite realizar la transferencia desde una dirección (la propietaria inicial) a otra (la propietaria final). Esta función está diseñada para que sólo pueda ejecutarla el propietario del token que se desea transferir u otra dirección previamente aprobada por el propietario.

```
    function _transfer(address from, address to, uint256 tokenId) interna
l virtual {
        require(NFTVehiculo.ownerOf(tokenId) == from, "ERC721: transfer o
f token that is not own");
        require(to != address(0), "ERC721: transfer to the zero address")
;

        _beforeTokenTransfer(from, to, tokenId);

        // Clear approvals from the previous owner
        _approve(address(0), tokenId);

        _balances[from] -= 1;
        _balances[to] += 1;
        _owners[tokenId] = to;

        emit Transfer(from, to, tokenId);
    }
```

Delegar la gestión del token con “approve()” y “setApproveForAll()”

Estas funciones permiten otorgar a una dirección diferente a la del propietario de los tokens el permiso de gestionar un token concreto o todos los que se poseen, respectivamente.

Suelen ser utilizadas para delegar gestiones en contratos de terceros y se deben ejecutar con cautela, otorgándole el permiso sólo a agentes de confianza.

```
    function _approve(address to, uint256 tokenId) internal virtual {
        _tokenApprovals[tokenId] = to;
        emit Approval(NFTVehiculo.ownerOf(tokenId), to, tokenId);
    }

    function setApprovalForAll(address operator, bool approved) public vi
rtual override {
        require(operator != _msgSender(), "ERC721: approve to caller");
```

```
        _operatorApprovals[_msgSender()][operator] = approved;
        emit ApprovalForAll(_msgSender(), operator, approved);
    }
```

Otras funciones

El estándar contiene otras funciones que sirven de ayuda a la hora de programar, para evitar vulnerabilidades y para dar mayor legibilidad al código.

También incluye otras cuyo fin es realizar llamadas de lectura para obtener fácilmente información de interés contenida en el contrato como, por ejemplo, quién es el propietario de cierto token, o qué direcciones han sido aprobadas para gestionarlo, etc.

Es importante recordar que, si durante el procesamiento de alguna función ocurriese algún error y no se pudiesen completar todas las operaciones, los cambios realizados se revertirían, por lo que no es posible que un error a mitad de procesamiento pueda dar lugar a situaciones inesperadas.

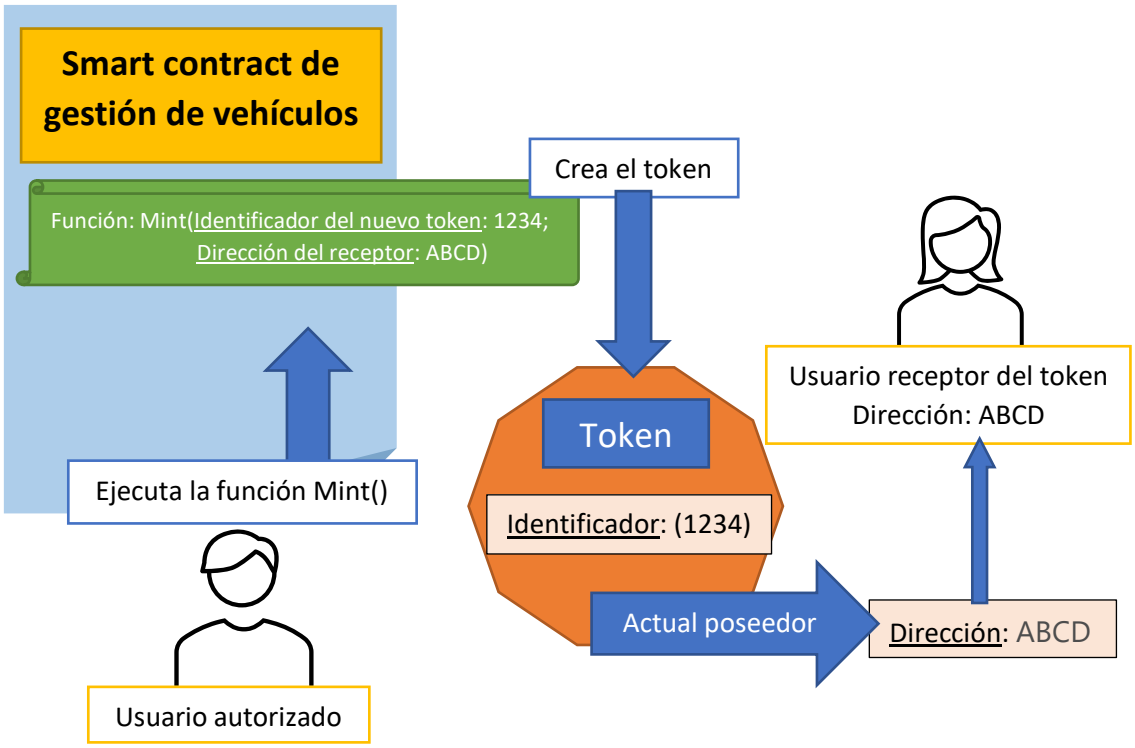

Figura 23. Proceso de creación de un token. Fuente: Elaboración propia.

2.1.3. El token como certificado de propiedad

Vincular un token no fungible con un vehículo es una excelente forma de demostrar ante terceros que se posee la *propiedad* del vehículo que éste identifica. La creación del token, mediante la función "Mint" debe proceder de algún agente que posea una autoridad reconocida sobre el producto o el mercado. En el caso de un vehículo ésta podría ser la compañía automovilística, la cual, crearía el token y se lo enviaría al cliente en el momento de la primera compra. El resto de los participantes en el mercado sería consciente de este hecho para los automóviles de esta marca y en situaciones en las que fuera necesario demostrar la propiedad del vehículo, el usuario podría aportar el token en vez de someterse a procesos administrativos más complejos.

También podría aportar la garantía de creación el gobierno, entregando una identidad digital a cada coche de la misma forma que lo hace con las matrículas, con la diferencia de que, utilizando la tecnología blockchain, no se podrían ni falsificar, ni duplicar.

En el momento de realizarse una venta de segunda mano, el comprador podría (y debería) exigir la transferencia del token en el momento de la transacción. De esta forma el comprador obtiene la garantía de que la procedencia del vehículo es legítima y no ha sido robado. Esto puede aumentar significativamente el valor de los coches de segunda mano.

Esta seguridad es especialmente importante en países en vías de desarrollo, dónde los robos son más frecuentes, los derechos de propiedad más difusos y su reventa en el mercado está bastante menos controlada que en países del primer mundo.

La incorporación del vehículo en la blockchain facilita además el acceso de éste a las finanzas descentralizadas. En el procedimiento de transferencia del token se puede incluir directamente el pago, de forma que ambos ocurran al mismo tiempo, como se va a mostrar a continuación.

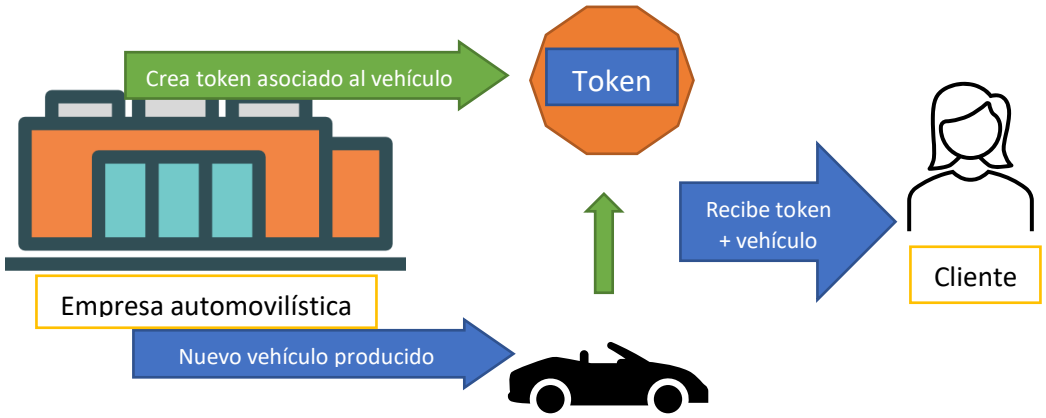

Figura 24. Emisión de un nuevo token asociado a un vehículo. Fuente: Elaboración propia.

2.2. Aplicación de compraventa

Una vez definida la propiedad del vehículo como la posesión del token que lo representa, se hace necesario transferir el token al mismo tiempo que se vende el vehículo en el mercado de segunda mano. Utilizando un Smart contract es posible realizar el intercambio dinero-token simultáneamente de forma que, sin la necesidad de depender de burocracia, mantendremos la confianza de que el pago va a ser realizado y el vehículo recibido por el comprador. Si alguna de las dos condiciones no se cumpliese, no se realizaría la operación.

La transacción del token dentro del contrato inteligente ejecuta en una simple operación todos los aspectos requeridos de un contrato de compraventa ya que en éste se incluyen:

- Estado del vehículo.
- Identidad del comprador, representada por su identificación en la blockchain.
- Identidad del vendedor.
- Fecha y hora.
- Datos bancarios, ahora sustituidos por el pago instantáneo en la blockchain.
- Documentación del vehículo.

Además, sería posible el pago automático de los impuestos requeridos por la operación. Todo ello sin necesidad de contratar a intermediarios, como gestorías, o delegar el intercambio a empresas especializadas en este negocio.

Este ahorro de costes conllevaría una reducción en el precio para el comprador, al mismo tiempo que el vendedor obtendría mayor retorno del que hubiese experimentado si delega la tarea de venta a terceros. Sin olvidar el ahorro de tiempo.

2.2.1. Esquema del proceso de compraventa

Juan tiene un vehículo en propiedad y desea realizar una venta a María. El vehículo de Juan tiene asociado un certificado de propiedad en forma de token, el cual recibió en el momento de la compra al concesionario. Ahora Juan va a aprovechar esta tecnología para realizar la venta del vehículo transfiriendo el certificado a María. Al mismo tiempo, María realiza el pago por el vehículo de forma automática, en el momento que se transfiere el token.

1. María le indica a Juan cuál es la dirección en la que quiere recibir el certificado de propiedad.

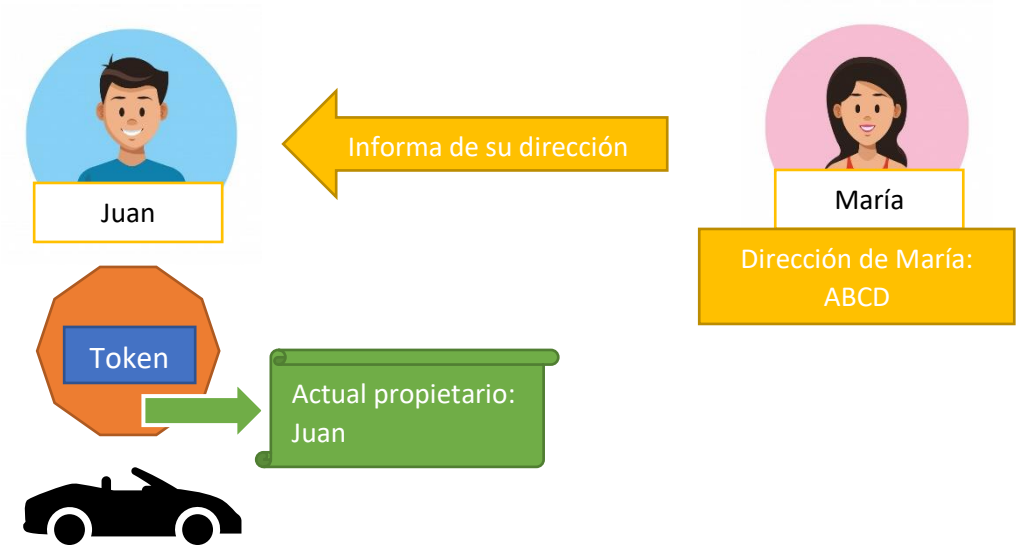

Figura 25. Venta, estado inicial. Fuente: Elaboración propia.

2. Juan activa la casilla de venta de su token, indicando la dirección de María y el precio de venta.

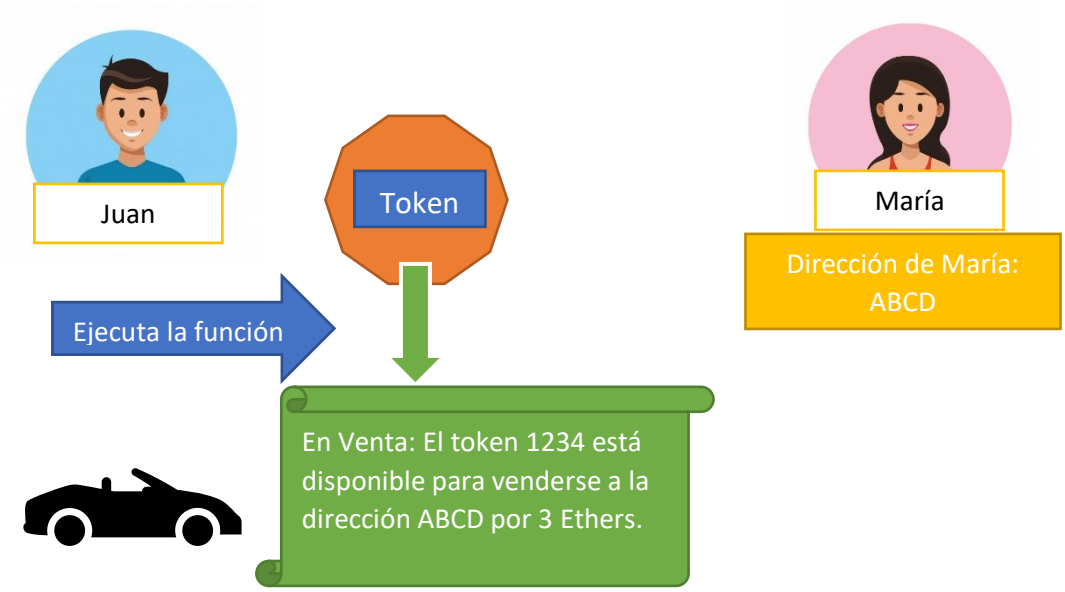

Figura 26. Token puesto en venta. Fuente: Elaboración propia.

En este punto aún no se ha realizado la transferencia del token y Juan puede cancelar la operación.

3. María ahora está autorizada a ejecutar la transferencia. Para ello debe ejecutar la función de Compra enviando el pago requerido por Juan.

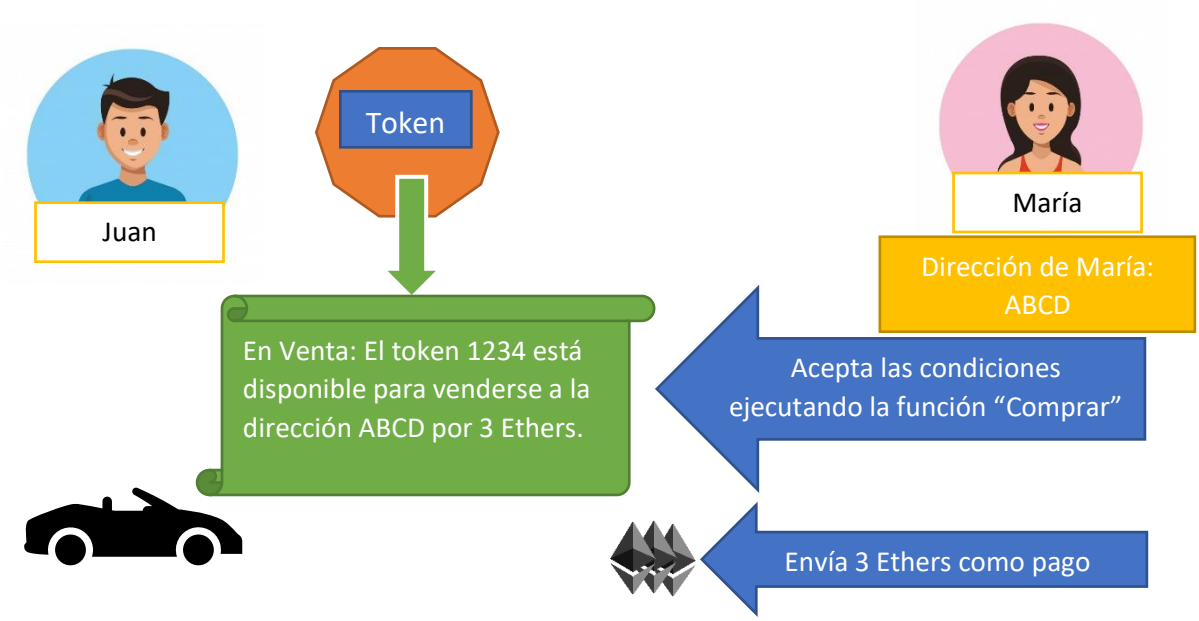

Figura 27. Ejecución de la compra. Fuente: Elaboración propia.

4. El proceso de compra finaliza. Juan recibe su pago y María obtiene el token del certificado de propiedad del vehículo.

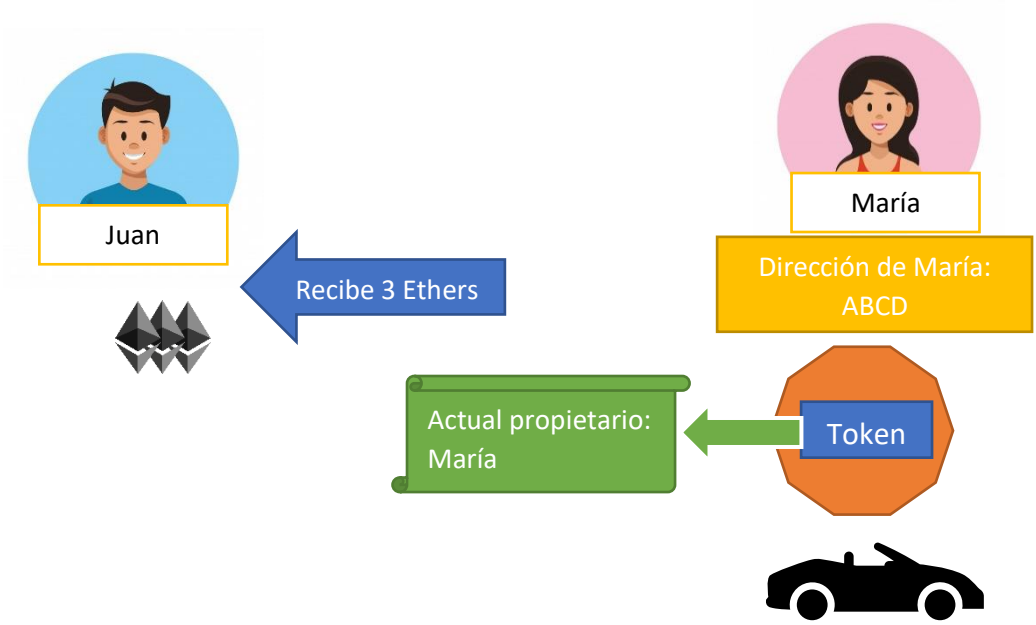

Figura 28. Venta, estado final. Fuente: Elaboración propia.

Si se envía una cantidad de dinero incorrecta o intenta comprarse el vehículo desde una dirección que no ha sido autorizada, la operación no se llevará a cabo.

2.2.2. Proceso de compraventa, programación del Smart contract.

a. Variables de estado de la venta

```
struct Pagos{
    bool _EstaEnVenta; //Predeterminado en false
    address _comprador;
    uint256 _precio;
    HistVentas[] _HistVentas;
}

struct HistVentas{
    address _Propietarios;
    uint256 _ValorDeVenta;
}

// mapping de token a Struct de pagos
mapping (uint256 => Pagos) private _Ventas;
```

Aquí se definen los parámetros necesarios para poder efectuarse el proceso de venta y los datos que se guardarán en memoria para poder acceder al histórico de transacciones fácilmente.

Se realiza un mapping en el que se asigna al identificador del token => Las variables de estado de venta de ese token.

Estas variables consisten en:

- Booleano *_EstaEnVenta*, el vendedor lo colocará en “true” si el token está listo para venderse.
- Dirección en la blockchain del comprador (*_comprador*).
- Precio de venta (*_precio*). En este caso se denominará en unidades de Ether.
- Hitorial de Ventas (*_HistVentas*). Vector en el que se guardará el histórico de propietarios que ha tenido el vehículo (*_Propietarios*) y el precio de venta (*_ValorDeVenta*).

Se podría pensar que como cada transacción queda registrada para siempre en la blockchain, no es necesario añadir un Historial de ventas que guarde esta información en el contrato. La realidad es que localizar una transacción concreta puede ser similar a buscar una aguja en un pajar. El hecho de utilizar un Estándar ERC-721 ayuda en esta tarea, ya que muchos servicios realizan un seguimiento de este tipo de tokens, pero aun así, es mucho más sencillo guardar directamente los datos en una variable del contrato para que el acceso sea rápido y sencillo.

b. Funciones del contrato

EnVenta(). Función que establece el estado del vehículo como disponible para la venta.

```
    function EnVenta(uint256 tokenId, address comprador, uint256 precio)
public returns(address, bool, uint256){
        require(_isApprovedOrOwner(_msgSender(), tokenId), "No eres el pr
opietario o no estas autorizado");     //Si es owner implica que existe _
exists() no necesario
        require(ownerOf(tokenId) != comprador, "No puedes venderte a ti m
ismo");
        _Ventas[tokenId]._comprador = comprador;
        _Ventas[tokenId]._EstaEnVenta = true;
        _Ventas[tokenId]._precio = precio * 1 ether;
        return(_Ventas[tokenId]._comprador, _Ventas[tokenId]._EstaEnVenta
, _Ventas[tokenId]._precio);
    }
```

Esta función solo es accesible por el propietario del token o alguna dirección aprobada por éste. En ella, el vendedor indicará la "matricula" del vehículo que quiere vender, la dirección del comprador y el precio de venta.

Al ejecutarse, aún no se realiza la venta, solo se define el estado de los parámetros de venta. Para completarse debe ser aceptada por el comprador.

EstaEnVenta.

Esta función es de sólo lectura. Tiene como fin poder leer los parámetros de la venta. Al introducir la identificación del vehículo devuelve el estado de la venta (los definidos en la función EnVenta.

```
    function EstaEnVenta(uint256 tokenId) public view returns(Pagos memor
y) {
        return(_Ventas[tokenId]);
    }
```

CancelarVenta.

```
    function _CancelarVenta(uint256 tokenId) internal virtual{
        _Ventas[tokenId]._comprador = address(0);
        _Ventas[tokenId]._EstaEnVenta = false;
        _Ventas[tokenId]._precio = 0;
    }

    function CancelarVenta(uint256 tokenId) public returns (string memory
) {
        require(_isApprovedOrOwner(_msgSender(), tokenId));
        _CancelarVenta(tokenId);
        return("Ya no esta a la venta");
    }
```

Si nos hemos equivocado al definir los parámetros de venta o, al final no se vende el vehículo, se puede cancelar utilizando esta función, devolviendo el estado de la venta al punto inicial.

Comprar(). Función que ejecuta el intercambio del token por dinero

```
    function Comprar(uint256 tokenId) public payable {
        require(_Ventas[tokenId]._comprador == _msgSender(), "No estas au
torizado como comprador"); //Solo si has sido designado comprador puedes
comprarlo
        require(msg.value == _Ventas[tokenId]._precio, "El valor de pago
no es correcto");

        address payable owner1 = payable(ownerOf(tokenId));
        uint256 pago = _Ventas[tokenId]._precio;

        _CancelarVenta(tokenId);
        _transfer(owner1, _msgSender(), tokenId);

        owner1.transfer(pago);
        _Ventas[tokenId]._HistVentas.push(HistVentas(_msgSender(), pago))
;
    }
```

Esta función será ejecutada por el comprador si está conforme con las condiciones de la venta. Evidentemente sólo el comprador del token puede acceder a esta función, el cual fue designado por el vendedor al ejecutar la función “EnVenta”.

El comprador debe introducir el identificador del vehículo que va a comprar e incluir en la transacción el pago exigido por el comprador. El ether pagado será enviado directamente a la dirección del vendedor y se transferirá la propiedad del token. Si el pago fuese inválido, la operación se revertiría y el intercambio token-ether no se llevaría a cabo.

Tras ello, el estado de la venta volvería a sus valores predeterminados (de 0) y se incluiría en el Historial de ventas del vehículo al nuevo propietario y el pago realizado.

2.3. Car-sharing

Car-sharing es un modelo de alquiler de vehículos que ofrece al usuario la posibilidad de alquilar su coche durante periodos cortos de tiempo. Es un servicio muy atractivo para personas que quieren hacer un uso ocasional del vehículo.

En la actualidad en España, los coches están de media aparcados el 97% del tiempo. Esto supone un desaprovechamiento muy importante de la utilidad del automóvil. Si se consiguiese una forma efectiva de hacer que usuarios particulares pudiesen alquilar durante periodos cortos de tiempo sus vehículos, se podría reducir el número de coches en circulación, el espacio dedicado a los aparcamientos en las ciudades y el consumo de recursos que se requieren para una producción tan sobredimensionada. [40] [41] [42]

Hoy en día, este tipo de servicios se ofrecen por empresas que disponen de una flota de vehículos que ponen a disposición del público, como Avancar o Car2Go. Sin embargo, su zona de actividad se limita a las ciudades más importantes, como Barcelona o Madrid, en España. Por otra parte, estas empresas ofrecen una solución al cliente que requiere de un vehículo ocasional introduciendo nuevas flotas de coches en las vías públicas, pero no aportan soluciones para los vehículos inactivos de particulares.

Solucionando este último problema han aparecido empresas como getaround, que permiten a usuarios individuales ofrecer sus coches en alquiler, de una forma similar a la que lo hace Airbnb con las viviendas. Sin embargo, depender de un intermediario que controle todo el servicio puede generar problemas si este acaba dominando el mercado e imponiendo comisiones demasiado altas, limitando el acceso a su aplicación o negándose a realizar colaboraciones con otras empresas que puedan ampliar o mejorar el servicio ya existente.

Para solucionar los problemas de monopolio que puedan surgir, la tecnología blockchain puede ofrecer una plataforma descentralizada y automatizada que brinde total transparencia sobre quién es el propietario del vehículo, a quién se ha alquilado y ejecute automáticamente el pago. Esta plataforma podría ser escalada y aprovecharse del “efecto red”, sin que un solo actor monopolizase esa red, ya que seguiría unas normas predeterminadas y sería de libre acceso. De esta forma también se favorecería la expansión a ciudades más pequeñas y pueblos.

En este trabajo se va a diseñar un contrato inteligente cuya finalidad es ofrecer un servicio de alquiler que resida en la blockchain. Todo propietario de un vehículo que esté vinculado al token que se ha diseñado previamente podría beneficiarse de este servicio, si le interesase, y todo potencial cliente podría acceder a él. Los servicios en la blockchain, al ser descentralizados, no necesitan depender de servidores de terceros ni de empresas intermediarias, lo que reduce los costes para el cliente y aumenta el beneficio para el propietario.

No sería estrictamente necesario integrar esta aplicación en el mismo contrato que genera el token. Gracias a la función “approve” se podría delegar la gestión de la información del token en el contrato de un tercero. Este tercer agente podría establecer una lógica de negocio en la que ofrece como servicio el diseño del Smart contract de car-sharing y cobrar automáticamente una comisión (muy pequeña) por cada uso que se haga del mismo.

2.3.1. Esquema del servicio de alquiler

Juan no va a utilizar su coche durante el fin de semana. Decide ponerlo a disposición de otros usuarios para aprovechar a ganar un dinero extra.

1. Juan establece en el Smart contract su deseo de alquilar su vehículo, de esta forma queda visible en la red. También establece el precio que quiere cobrar por hora

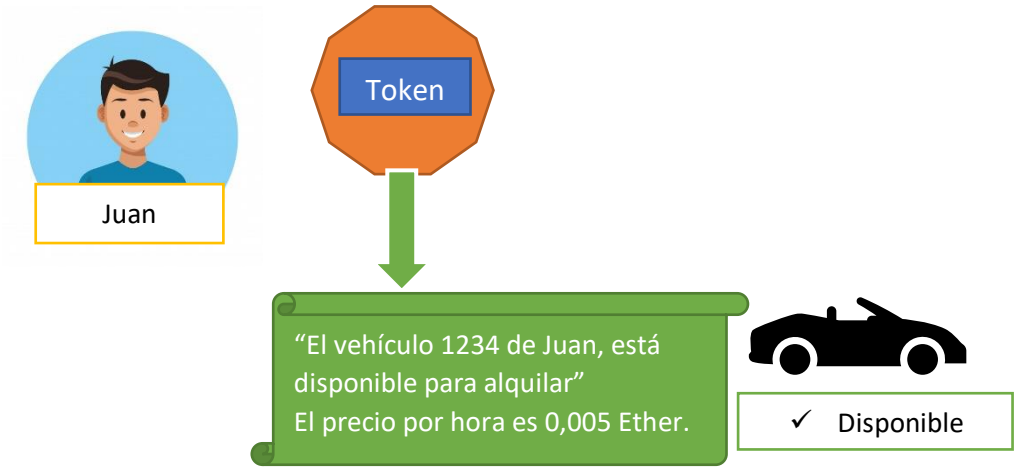

Figura 29. Puesta en alquiler. Fuente: Elaboración propia.

Juan en cualquier momento puede cancelar la puesta en alquiler o cambiar el precio si así lo desea.

2. María quiere alquilar el vehículo. Para ello, utilizando la función de Alquiler, selecciona el vehículo, las horas que lo va a utilizar y envía el pago.

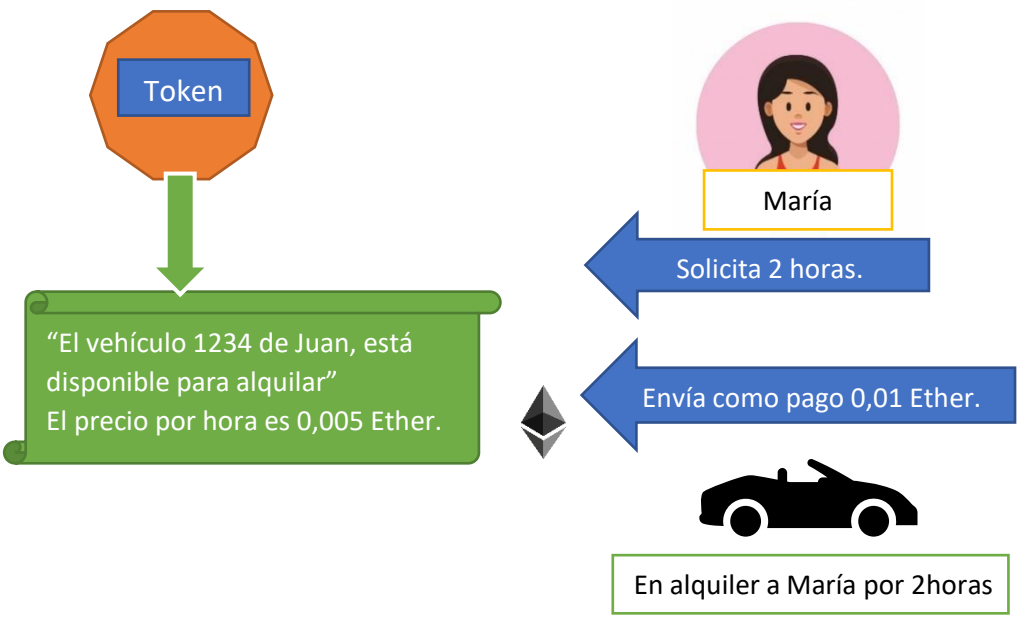

Figura 30. Solicitud de alquiler. Fuente: Elaboración propia.

En esta situación nos enfrentamos a 2 problemas:

El primero es el acceso remoto al vehículo. Para ello sería necesario un sistema que pueda abrir las puertas automáticamente en el momento que detecte que se da un evento de alquiler en la blockchain. Algunas marcas de coches ya incluyen la posibilidad de abrir las puertas utilizando el smartphone, el sistema sería similar solo que estando conectadas a la blockchain.

El segundo, es que es necesario conocer la identidad del usuario que alquila el vehículo, por si ocurriese cualquier problema. Por ello, se puede incluir como condición en el smartcontract, que la dirección (en principio anónima) que quiere alquilar el vehículo, haya pasado a través de algún servicio de verificación de su identidad, llevado a cabo por un seguro privado, la DGT u otra empresa que se pueda dedicar a ello.

3. Al terminar el fin de semana, Juan cancela la oferta del servicio. En su dirección de la blockchain ha recibido los pagos obtenidos a lo largo del fin de semana.
   Toda la información sobre la actividad en el alquiler queda registrada de forma permanente en la blockchain.

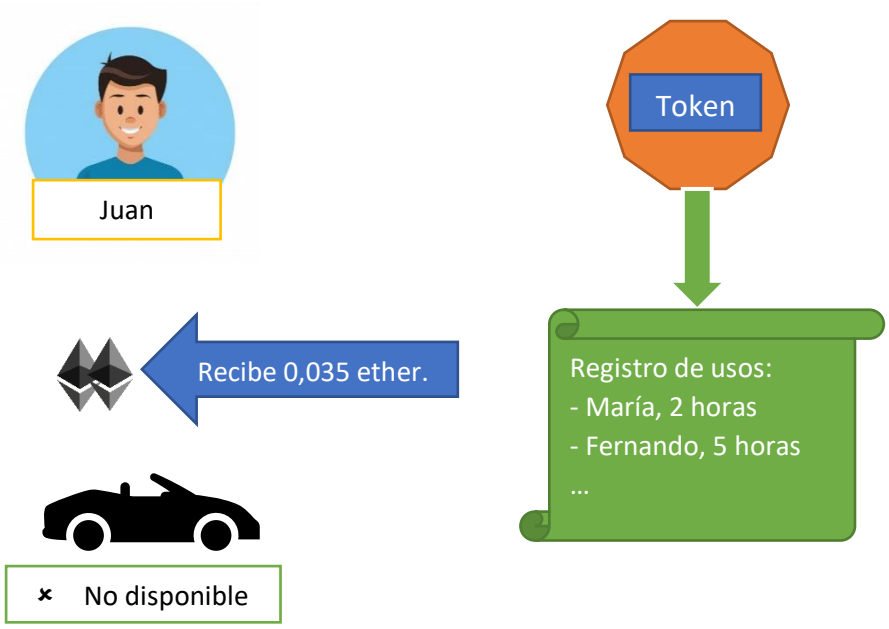

Figura 31. Fin del alquiler. Fuente: Elaboración propia.

En este caso, el pago se realiza en la moneda nativa de la red Ethereum, el Ether. Es la forma más sencilla. Sin embargo, existirían formas de hacer que el pago se pudiese realizar en una moneda estable vinculada al euro o el dólar.

2.3.2. Servicio de alquiler utilizando Smart contracts

a. Variables de estado de Alquiler

```
    struct RegistroProp {
            address _direccion;
            uint256 _HoraInic;
            uint _horas
        }

    struct Alquiler {
        bool _disponible;           //Marca si el coche esta disponible p
ara poder alquilarse
        bool _alquilado;            //Marca si el coche está alquilado en
 este momento
        uint256 _PrecioHora;
        uint256 _NumUsuarios;       //Num de usuarios que lo han utilizad
o
        RegistroProp[] _RegistroProp;
    }

    mapping (uint256 => Alquiler) private _Alquiler;        //Mapping del
 vehículo a sus datos de alquiler
```

Se definen los parámetros necesarios para la gestión de la información en el contrato de alquiler. A cada identidad del vehículo se la relaciona a través de un mapping con las variables de estado de alquiler.

El Alquiler contiene los siguientes datos.

- Un booleano que determina si el vehículo está disponible para ser alquilado y puede incluirse en esta aplicación. Será el propietario el que, a través de una función lo pondrá "true". (*_disponible*).
- Otro booleano que informará de si el vehículo está ya alquilado en este momento (*_alquilado*).
- El precio del servicio (*_PrecioHora*). Podría establecerse en minutos, horas, días… En este trabajo se determina en horas.
- Numero de usuarios. Almacena la cantidad de veces que un determinado vehículo ha sido alquilado (*_NumUsuarios*). La información del histórico de usuarios se almacena en un vector, por lo que es necesario llevar la cuenta de cuantas entradas tiene este vector.
- (*_RegistroProp*). Es un vector que almacena la información de los usuarios que han utilizado el servicio de alquiler. Por cada cliente se almacenan dos parámetros:
    - La dirección del usuario (*_direccion*).
    - La hora y fecha en la que comienza el alquiler (*_HoraInic*). El valor real que contiene esta variable es la marca de tiempo del bloque de la blockchain en el que se ejecuta la transacción en el momento del alquiler.
    - La cantidad de horas por las que se contrata el servicio (*_horas*).

Además de la vinculación (Identidad del token) => (Datos del alquiler), se hace necesario establecer una relación entre un determinado usuario y su actividad con el contrato de alquiler. Para ello, también se crea un mapping que los relaciona de la forma: (Dirección del usuario) => (Historial del usuario).

```
    struct Registros {
        uint256 _tokenId;
        uint256 _numRegistro;
    }

    struct Historial {
        uint256 _numUsos;
        Registros[] _Registros;
    }

    mapping (address => Historial) private _Historial;          //Mapping
para guardar la actividad de un usuario
```

El historial del usuario del contrato de alquiler contiene los siguientes datos:

- El número de veces que el usuario ha alquilado el vehículo (*_numUsos*).
- Un registro con la información de cómo ha sido su interacción (*_Registros*). Contiene:
    - La identidad de la serie de vehículos que han alquilado (*_tokenId*).
    - La posición en la que ha quedado guardada su información en el registro de ese vehículo (*_numRegistro*). Con este número se puede acceder a la actividad del usuario en el (*_RegistroProp*) que se ha visto antes.

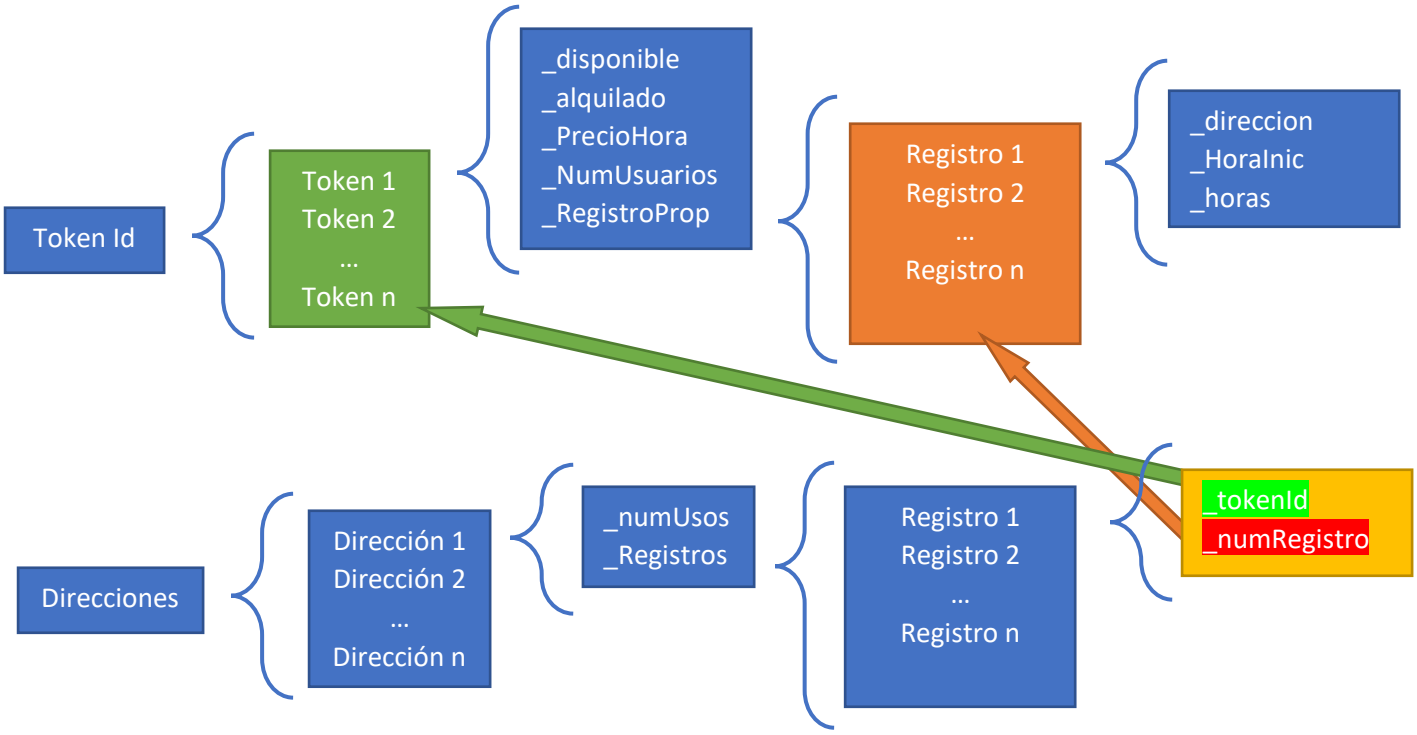

Figura 32. Esquema del árbol de registros en el servicio de alquiler. Fuente: Elaboración propia.

b. Funciones del contrato

PonerEnAlquiler

Similar a la función "EnVenta", el propietario del vehículo establece el estado del vehículo como disponible para alquilar y determina el precio por hora.

```
    function PonerEnAlquiler(uint256 tokenId, uint256 PrecioHora) public
{
        require(_isApprovedOrOwner(_msgSender(), tokenId), "No eres el pr
opietario o no estas autorizado");

        _Alquiler[tokenId]._disponible = true;
        _Alquiler[tokenId]._PrecioHora = PrecioHora * 1 ether;
    }
```

A diferencia de la función "EnVenta", aquí no se especifica la dirección de los usuarios que pueden acceder al alquiler, se permite el acceso libre.

RetirarDelAlquiler

En la misma línea que en el contrato de venta, se establece una función para retirar el vehículo del alquiler.

```
    function RetirarDelAlquiler(uint256 tokenId) public {
        require(_isApprovedOrOwner(_msgSender(), tokenId), "No eres el pr
opietario o no estas autorizado");

        _Alquiler[tokenId]._disponible = false;
        _Alquiler[tokenId]._PrecioHora = 0;         //Quiza instruccion i
nnecesaria
    }
```

DatosDeAlquiler

Función de sólo lectura para comprobar el estado del alquiler de un vehículo introduciendo su identidad.

```
    function DatosDeAlquiler(uint256 tokenId) public view returns(Alquile
r memory) {
        require(_exists(tokenId), "El Id no existe");
        return(_Alquiler[tokenId]);
    }
```

Alquilar

Esta función va a ser utilizada por un cliente que desee alquilar el vehículo. Se apoya en dos funciones secundarias que deben devolver "true" para que la función alquilar se ejecute:

_sePuedeAlquilar: Que devuelve "true" en el caso de que el vehículo esté disponible para alquilar.

_FinDelAnterior: Que devuelve "true" si el alquiler anterior al que ahora se pretende ejecutar ha finalizado, es decir, que se han consumido las horas contratadas.

```
    function _sePuedeAlquilar(uint256 tokenId) internal view virtual retu
rns(bool) {
        return(_Alquiler[tokenId]._disponible);
    }
```

```
    function _FinDelAnterior(uint256 tokenId) internal view virtual retur
ns(bool) {          //Nos dice si se ha terminado el tiempo del alquiler a
nterior (En el caso de que no hayamos puesto confirmacion)
        uint256 UltimoUser = _Alquiler[tokenId]._NumUsuarios - 1;
        uint256 HoraInic = _Alquiler[tokenId]._RegistroProp[UltimoUser]._
HoraInic;
        uint256 HoraFin = HoraInic + _Alquiler[tokenId]._RegistroProp[Ult
imoUser]._horas * 1 hours;
        bool HaTerminado;

        if (block.timestamp > HoraFin) {        //Si es la primera vez, b
lock.timestap siempre sera > HoraFin = 0
            HaTerminado = true;
        }
        else {
            HaTerminado = false;
        }
        return (HaTerminado);
    }
```

Para ejecutar la función, el cliente sólo tiene que introducir la Identidad del vehículo que quiere alquilar y adjuntar el pago en Ether que requiere el propietario. Cuando todas las condiciones se cumplen, la función "*Alquilar*" incluye los datos del cliente en el registro (RegistroProp[]) y transfiere el pago desde el cliente al propietario.

```
    function Alquilar(uint256 tokenId, uint256 horas) public payable {
        require(_exists(tokenId), "El Id no existe");
        require(_sePuedeAlquilar(tokenId), "El vehiculo no esta en alquil
er");
        if (_Alquiler[tokenId]._NumUsuarios != 0) {
            require(_FinDelAnterior(tokenId), "El vehiculo sigue en alqui
ler");
        }

        uint256 pagoTotal = _Alquiler[tokenId]._PrecioHora * horas;
        require(msg.value == pagoTotal, "El valor pagado es incorrecto");
        _Alquiler[tokenId]._NumUsuarios += 1;
        uint256 NumUsuarios = _Alquiler[tokenId]._NumUsuarios - 1;  //Par
a que comience con el 0

        _Alquiler[tokenId]._alquilado = true;
        _Alquiler[tokenId]._RegistroProp.push(RegistroProp(_msgSender(),
block.timestamp, horas));

        _GuardarUsuario(tokenId, _msgSender(), NumUsuarios);

        address payable owner1 = payable(ownerOf(tokenId));
        owner1.transfer(pagoTotal);

    }
```

La función *_GuardarUsuario* guarda los datos del Usuario en el "Historial[]"

```
    function _GuardarUsuario(uint256 tokenId, address user, uint256 numRe
gistro) internal virtual {
        _Historial[user]._numUsos += 1;
        _Historial[user]._Registros.push(Registros(tokenId, numRegistro))
;
    }
```

2.4. Pasaporte digital

Un pasaporte digital de un vehículo es un libro que guarda todos los eventos que éste ha experimentado a lo largo de su vida útil.

En el momento de realizar la venta de un coche en el mercado de segunda mano, preguntas como "¿Cuántos kilómetros ha recorrido el coche?, ¿Ha tenido algún accidente? o ¿Qué piezas han sido sustituidas?" surgen en el comprador.

Figura 33. Esquema básico del pasaporte digital de un vehículo. Fuente: Elaboración propia.

Dar una respuesta clara y confiable a estas preguntas generará una confianza mayor al interesado y como consecuencia, el valor al que se podrá vender un coche en buenas condiciones será mayor.

Para que este pasaporte digital sea posible, es necesario que pueda ser fácilmente accesible, tanto para leerlo como para escribir en él, y todos los agentes participantes puedan tener la confianza en que la información que recoja no pueda ser falsificada o reescrita.

La tecnología blockchain ofrece una respuesta a este problema, por ser descentralizada, es decir, que es fácilmente accesible por todos los agentes; e inmutable, su información no puede cambiarse a posteriori.

Que este pasaporte digital esté guardado en la blockchain tiene otras ventajas:

1. Debemos recordar que la blockchain es una base de datos, solo que descentralizada. Esto hace que no dependa de ningún servidor central, sino que la información está distribuida. Por ello, no existe el riesgo de que el servidor que almacena la información se caiga, o la empresa que la guarde deje de ofrecer el servicio en el futuro.

2. Se ha hablado anteriormente de los Smart contracts. En la blockchain la forma en la que se decide quién y cómo se guardan los datos de este pasaporte digital, queda regida por las condiciones que imponga el Smart contract que lo gestione. Por tanto, cualquier

agente tiene la posibilidad de escribir sobre este pasaporte digital, pero el propietario tiene total control sobre esta actividad.

3. Como consecuencia del punto anterior, la descentralización de la blockchain permite que cualquier agente pueda integrarse fácilmente en las dinámicas de registro de datos. Una de ellas, podría ser vincular las piezas que se incorporan al vehículo, con contratos externos en los que haya quedado registrada la trazabilidad de estos componentes, generándose así sinergias entre diferentes participantes.

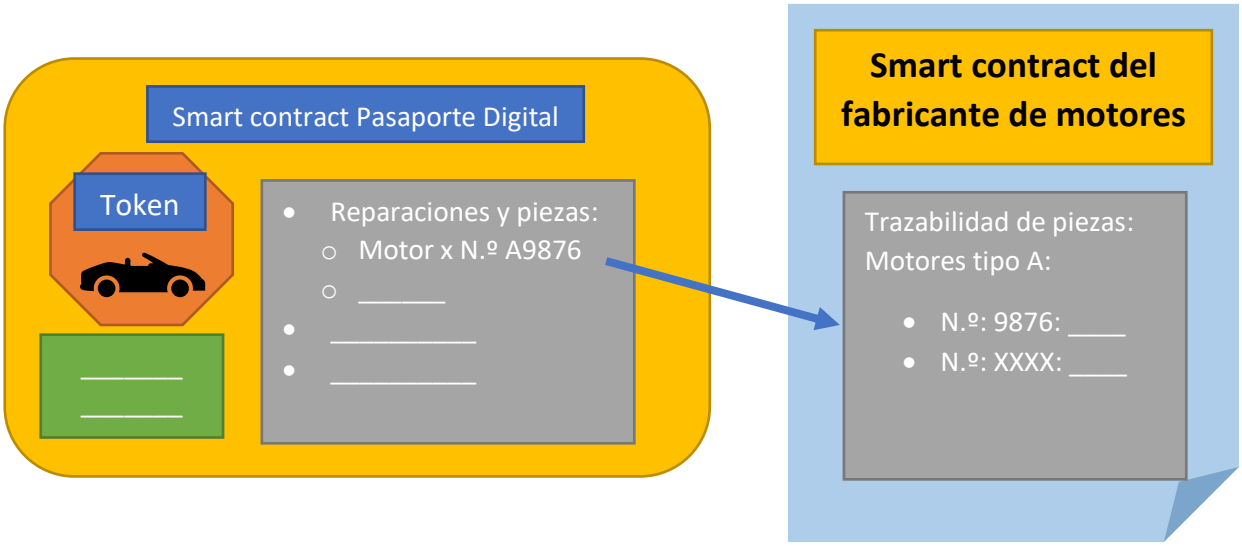

Figura 34. Ejemplo de conexión entre dos smart contracts. El pasaporte digital y la trazabilidad del fabricante de la pieza. Fuente: Elaboración propia.

Disponer de este historial de eventos que es el pasaporte digital, también puede incentivar a las empresas de seguros para que ofrezcan descuentos especiales ya que tendrán más información sobre el vehículo al que ofrecen el servicio.

En este trabajo, se va a proponer un pasaporte digital que registre la siguiente información:

- Recambios: Se guardará información relacionada con piezas del vehículo que han sido sustituidas.
- Eventos: Aquí se guardarán eventos relevantes que le hayan ocurrido al vehículo como pueden ser accidentes, cambios de seguros, cambios de matrícula, modificaciones especiales…
- Revisiones: Se guardan revisiones realizadas al vehículo, como puede ser la ITV.

La estructura de la información se describirá en detalle en el siguiente apartado.

a. Estructura de los datos del pasaporte digital

La información se va a guardar en un vector diferente por cada tipo de actividad. Las actividades que se tienen en cuenta son recambios de piezas, eventos y revisiones. Cada vector es un histórico de variables tipo struct en el que se incorporan los datos más relevantes de cada actividad registrada.

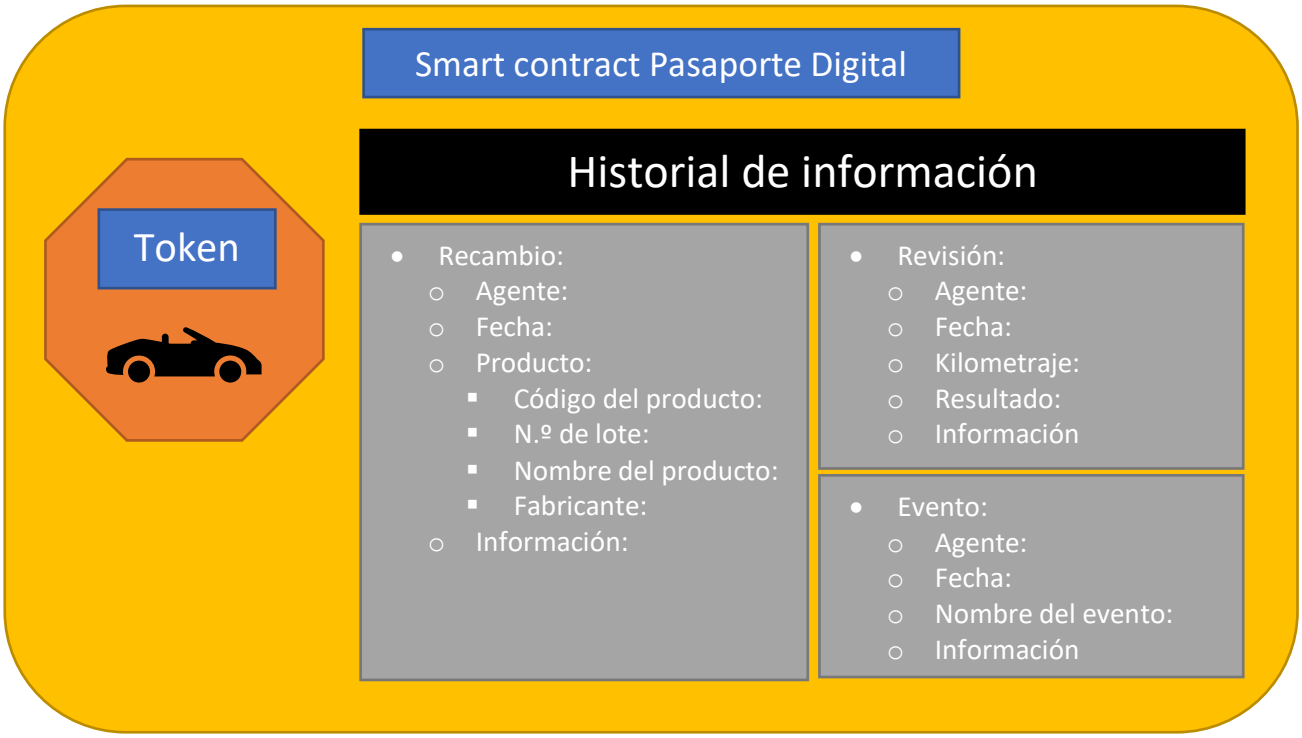

Figura 35. Estructura de la información en el pasaporte digital. Fuente: Elaboración propia.

Hay tres elementos que van a compartir los tres tipos de actividades.

- *_agente*:
  Guardará la dirección del usuario que escriba los datos de la actividad. Este agente puede ser un taller, un seguro, el propio usuario o alguna institución pública. Las direcciones de la blockchain, en principio, no están relacionadas con ningún agente en la vida real, pero la empresa que realice la escritura de la información debería, en este contexto, mostrar abiertamente al público qué dirección está utilizando para realizar este servicio. Así, un lector futuro podría comprobar la autoridad que pueda tener el usuario que realizó el registro.
- *_fecha*:
  La fecha del momento en el que se guardan los datos.
- *_info*:
  Es una variable que guarda texto. La idea es que el agente que realiza el registro pueda incluir información adicional. Es recomendable ser escueto ya que escribir mucho texto tiene un coste en gas. Si se desea incluir mucha información las mejores alternativas son:

- Incluir la dirección de otro Smart contract:
  Es posible que la empresa que realice el registro también realice otras actividades en la blockchain y cuente con un Smart contract propio donde amplía la información. Aquí se podría incluir esa dirección.
- Vincular un documento y escribir su hash:
  Si se desea que la información sea privada, otra alternativa es que la empresa redacte un informe propio y que lo guarde en su propia base de datos, de forma que sólo pueda verla el propietario del vehículo si así lo solicita. A este documento se le realizará la función "hash" y se escribirá el resultado en esta variable. En un futuro, en el momento de venta, el propietario podrá solicitar este documento y demostrar que es original mostrando que al realizar la función hash sobre éste, se obtiene el mismo resultado que en su momento se registró en la blockchain.

```
//Recambio
struct RecambioPieza {
    address _agente;        //Agente que realiza la operacion
    uint256 _fecha;         //Fecha de la operacion
    Producto _producto;     //Producto, pieza incorporada
    string _info;           //Informacion adicional
}

struct Producto {
    uint256 _productCode;       //Codigo de la pieza
    uint256 _productBatch;      //Número de lote o unidad
    string _productName;        //Nombre de la pieza
    address _productOwner;      //Fabricante
}
```

Los datos contenidos en la actividad de recambio incluyen además información sobre el producto. La mejor forma de identificar la pieza es mediante su número de serie, su número de lote y su nombre. Además, si la empresa que vende la pieza realiza la trazabilidad de ésta también mediante tecnología blockchain, se incluye la posibilidad de incluir la dirección del contrato sobre el que le ha dado seguimiento.

```
//Evento
struct Evento {
    address _agente;
    uint256 _fecha;
    string _eventName;
    string _info;
}
```

El struct del evento incluye adicionalmente la posibilidad de incluir el nombre de éste.

```
    //Revision
    struct Revision {
        address _agente;
        uint256 _fecha;
        uint256 _kilometraje;
        string _resultado;
        string _info;
    }
```

Por último, la revisión incluye un string en el que escribir el resultado de esta revisión y la posibilidad de incluir la lectura del odómetro en ese momento.

El fraude en el odómetro cuesta a los compradores estadounidenses más de mil millones de dólares al año. [43]

En España, desde el año 2014 la ITV está obligada a anotar la cifra del cuentakilómetros, esta información podría ser trasladada a la blockchain de una forma más habitual. [44]

Por otra parte, no todos los países llevan un seguimiento de estos datos. Contar con un histórico disponible en la blockchain puede aumentar la confianza de los compradores y el valor del vehículo en el mercado de segunda mano.

```
    //Mappings del tokenId a sus historiales de recambios, eventos y revi
siones
    mapping (uint256 => RecambioPieza[]) _RecambioPieza;
    mapping (uint256 => Evento[]) _Evento;
    mapping (uint256 => Revision[]) _Revision;
```

Por último, se relacionan estos vectores con la identidad del vehículo al que pertenecen mediante el uso de mappings.

b. Funciones del contrato

La utilidad de las funciones es básicamente habilitar la lectura y la escritura de la información. Pero para poder limitar quién puede escribir y quién no, hay que realizar unas funciones que entreguen los permisos. A estas funciones solo tendrá acceso el propietario del vehículo o la persona que haya delegado para la gestión mediante la función “approve”.

PermisoUnico

```
    function PermisoUnico(uint256 tokenId, address agente) public{
    //La funcion permite dar permisos de escritura en el PD una vez
        require(_isApprovedOrOwner(_msgSender(), tokenId));
        _permisoUnico[tokenId] = agente;
    }
```

Esta función entrega un permiso a la dirección elegida una sola vez.

PermisoGeneral

```
    function PermisoGeneral(uint256 tokenId, address agente, bool aprobad
o) public{         //La funcion permite dar permisos de forma indefinida
        require(_isApprovedOrOwner(_msgSender(), tokenId));
        _aprobadoGeneral[agente][tokenId] = aprobado;
    }
```

Como su nombre hace intuir, esta función entrega permisos para que el agente pueda escribir siempre que lo desee. La propia función permite que el propietario del vehículo cancele este privilegio cuando desee.

_PuedeEscribir

```
    function _PuedeEscribir(address agente, uint256 tokenId) internal vir
tual returns(bool) {           //Funcion de ayuda para permisos
        require(_exists(tokenId), "El vehiculo no existe");
        bool puede = (_permisoUnico[tokenId] == agente || _aprobadoGenera
l[agente][tokenId] || _isApprovedOrOwner(agente, tokenId));

        if (_permisoUnico[tokenId] == agente) {
            _permisoUnico[tokenId] = address(0);
        }

        return(puede);
    }
```

Esta función interna comprueba si el agente que realica el registro tiene permisos de escritura. Además, retira el permiso único a la dirección correspondiente si es la que está llamando a la función.

Las direcciones con permisos quedan guardadas y relacionadas con el vehículo en el que pueden escribir mediante el uso de mappings.

```
    //Mapping aprobados para escribir en el PD
    mapping (address => mapping (uint256 => bool)) private _aprobadoGener
al;        //Mapping de agente => tokenId => true/false
    mapping (uint256 => address) private _permisoUnico;
           //tokenId => direccion agente, cambia a 0 cuando el agente eje
cuta la operacion
```

Funciones de lectura y escritura

A continuación se muestran las funciones de escritura y lectura de los vectores Recambio[], Evento[] y Revision[]. Cada vez que se utiliza una función de escritura, ésta añade una nueva entrada en el vector correspondiente, antes de ello, comprueba si la persona que llama a la función _PuedeEscribir.

```
    function ARecambio(uint256 tokenId, uint256 productCode, uint256 prod
uctBatch, string memory productName, address productOwner, string memory
info) public {
        require(_PuedeEscribir(_msgSender(), tokenId));
        Producto memory _producto1;
        _producto1._productCode = productCode;
        _producto1._productBatch = productBatch;
        _producto1._productName = productName;
        _producto1._productOwner = productOwner;
        _RecambioPieza[tokenId].push(RecambioPieza(_msgSender(), block.ti
mestamp, _producto1, info));
    }

    function VerRecambios(uint256 tokenId) public view virtual returns (R
ecambioPieza[] memory) {
        return (_RecambioPieza[tokenId]);
    }

    function AEvento(uint256 tokenId, string memory eventName, string mem
ory info) public {
        require(_PuedeEscribir(_msgSender(), tokenId));
        _Evento[tokenId].push(Evento(_msgSender(), block.timestamp, event
Name, info));
```

```
    }

    function VerEventos(uint256 tokenId) public view virtual returns (Eve
nto[] memory) {
        return(_Evento[tokenId]);
    }

    function ARevision(uint256 tokenId, uint256 kilometraje, string memor
y resultado, string memory info) public {
        require(_PuedeEscribir(_msgSender(), tokenId));
        _Revision[tokenId].push(Revision(_msgSender(), block.timestamp, k
ilometraje, resultado, info));
    }

    function VerRevisiones(uint256 tokenId) public view virtual returns (
Revision[] memory) {
        return(_Revision[tokenId]);
    }
```

3. El modelo de negocio asociado al servicio de post venta.

3.1. Implementación como modelo de negocio

La compraventa, el alquiler y el pasaporte digital son servicios proveídos por este contrato en la blockchain y aportan un valor que puede ser ofrecido y capitalizado por distintos agentes. El pasaporte digital es en el que confluyen los intereses de una mayor diversidad de participantes.

3.1.1. Agentes participantes

a. Empresa automovilística

La empresa automovilística es la encargada de emitir un token por cada vehículo puesto en circulación. Ofrece la garantía de que el vehículo existe y tiene las características iniciales mostradas en el Smart Contract.

Al ser la generadora de los tokens y darles su respaldo inicial, está habilitando a sus compradores a participar en los servicios de postventa antes enunciados. Hacer esto, se materializa en mayor interés por parte del cliente a la hora de comprar sus vehículos ya que se beneficiará del servicio de posventa.

b. Propietarios

Beneficiarios directos del servicio. En primer lugar, tendrán mayores facilidades a la hora de acceder al mercado de segunda mano gracias al servicio de compraventa, también podrán obtener ingresos de su vehículo gracias al servicio de alquiler y finalmente, el valor de su coche aumentará en el mercado de segunda mano gracias al pasaporte digital.

c. Empresas de seguros

Ganan acceso a información de gran importancia a la hora de ofrecer un seguro a sus clientes. Al poder comprobar los antecedentes del vehículo fácilmente podrán ofrecer servicios personalizados y reducir las cuotas de sus usuarios. En este sentido, los intereses del cliente convergen con los de la aseguradora, estando el cliente más dispuesto a incluir información en el pasaporte digital de su vehículo, al mismo tiempo que a la aseguradora le interesa incentivar que lo haga.

d. Talleres

Los talleres serán en gran medida los responsables de escribir información en el pasaporte, ya que es el lugar dónde se realizan las reparaciones y las revisiones. El cliente puede tener la preferencia de ir a un taller que realice el registro en la blockchain respecto a otros que no lo hagan.

En un modelo de pasaporte digital en el que cualquier actor puede escribir si así lo decide el usuario, podría darse el caso de que ciertos talleres incluyesen información falsa en el pasaporte digital con el fin de beneficiar al cliente con un pasaporte mejor adornado. Esta posibilidad tiene como consecuencia que existan talleres cuyos registros tengan mayor o menor confianza para el público. Si se empezasen a observar comportamientos desleales por parte de ciertos talleres, se perdería la confianza en sus registros y, los clientes podrían no querer visitarlos más, en favor a otros que si tengan una mejor reputación.

Por ello, el conflicto de intereses termina equilibrándose por sí mismo, ya que el incentivo de un taller será ofrecer el mejor servicio posible para tener la mayor confianza posible dentro de su entorno.

Los talleres también pueden colaborar con los fabricantes de piezas para darle seguimiento a la trazabilidad de los componentes, lo cual aumentaría la confianza general de la información registrada en el pasaporte digital.

e. Gobiernos

Los servicios de postventa están dispuestos para poder ser ofrecidos de forma privada. Sin embargo, es necesario el visto bueno del gobierno para que puedan ejecutarse de forma más efectiva. Este es el caso del concepto de propiedad.

A la hora de realizar la compraventa, se está utilizando el token como documento que acredita a su poseedor como propietario del vehículo.

Esta definición de propiedad tiene lagunas regulatorias y muchos países están dando pasos en la dirección de definir una normativa clara. Es necesaria la acción del gobierno para que este tipo de interacciones entre personas sean una realidad.

Por otra parte, las administraciones también pueden ver en el pasaporte digital una herramienta para llevar un control de los vehículos, pudiéndoles asignar una matrícula y comprobar rápidamente su historial sin riesgo de falsificaciones o fraudes. También se podrían implementar servicios externos donde para realizar trámites administrativos relacionados con el vehículo, se pueda acceder a través del token, de forma similar a como se hace hoy en día con el DNI electrónico.

Hay dos formas de implementar este servicio. Una es a través del interés directo de una empresa automovilística y la otra, de forma independiente por los diseñadores del contrato y sus accionistas.

3.2. Implementación por parte de una compañía automovilística

En este caso, la empresa que promueve el servicio es la empresa automovilística. La implementación blockchain es ofrecida como un servicio adicional que viene incluido con la venta del vehículo. La empresa, considerando que este valor diferencial la hará destacar sobre su competencia, se encarga de promover este servicio de postventa exclusivo.

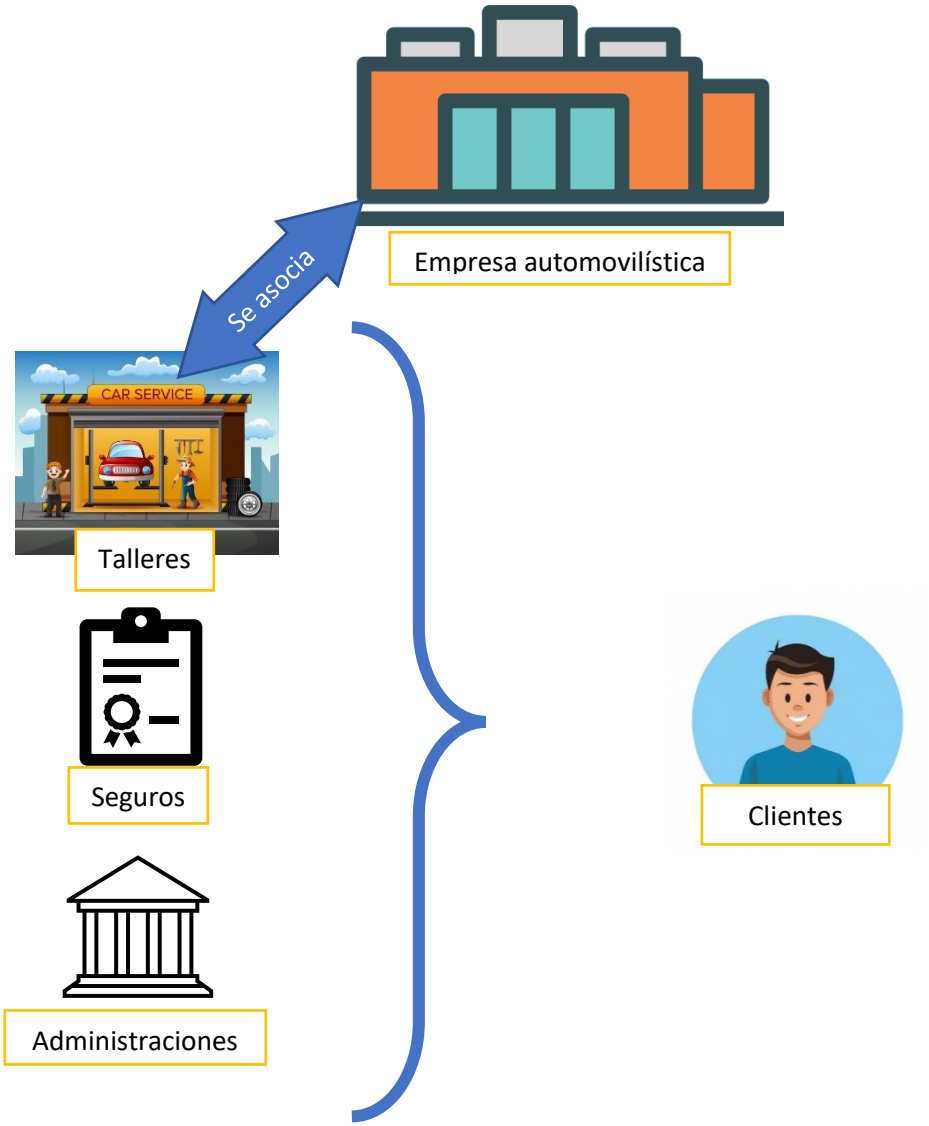

Figura 36. Relaciones entre agentes. Fuente: Elaboración propia.

Siguiendo este modelo, la empresa de automóviles se asocia con talleres, seguros y administraciones para que participen en el proyecto de forma activa. Con estas asociaciones se pretende dar el impulso inicial que necesitan este tipo de proyectos para acabar sosteniéndose por sí mismos en el largo plazo, ya que todos los participantes obtienen ventajas.

Una forma despliegue apropiada, consistiría en comenzar a ofrecer el servicio de blockchain en modelos de gama alta o de unidades limitadas, para posteriormente y, a medida que se confirmase la aceptación por parte de los clientes, ir incorporándolo a vehículos de menor precio.

3.2.1. Plan estratégico

El plan para introducir en el mercado los servicios de blockchain constará de tres fases, en las que se comenzará a ofrecer el servicio a un pequeño grupo de clientes para ir ampliando la oferta hacia el mercado de masas.

Este crecimiento en el alcance será conducido desde vehículos exclusivos hacia vehículos de gamas de menor precio, al mismo tiempo que los servicios ofertados van creciendo en su complejidad y en la necesidad de que existan ya redes de usuarios que los utilicen de forma activa.

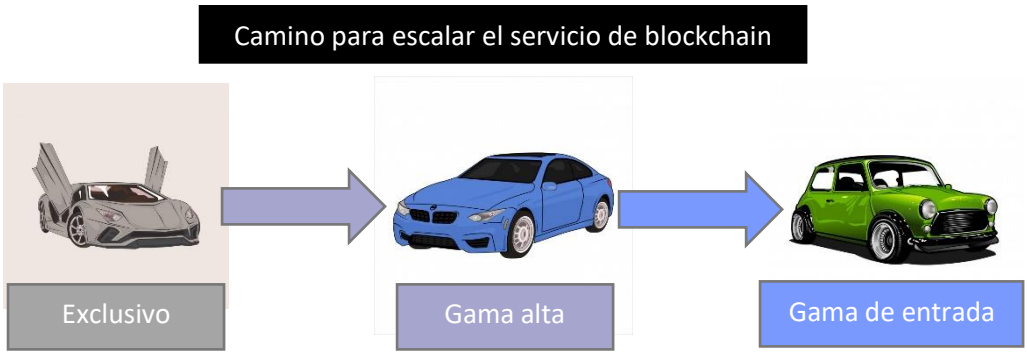

Figura 37. Camino a seguir en la implementación del servicio. Fuente: Elaboración propia.

	Gama del vehículo	Número de clientes	Servicios incluidos
Fase 1	Exclusiva	Pocos	Token + Compra/Venta
Fase 2	Se incluye la gama alta	Medios	Se añade el pasaporte digital
Fase 3	Se incluye la gama de entrada	Muchos	Se añade el servicio de alquiler

A continuación, se detallan las fases y el porqué de esta decisión.

Fase I. Vehículos exclusivos.

En esta primera fase se ofrecería a los compradores de vehículos de lujo el token como un elemento de coleccionista. La idea de este modelo es asociar tokens únicos a artículos de lujo para darles exclusividad, garantía de propiedad y originalidad.

Ventajas:

- Al dirigirse a un público con alto poder adquisitivo, se puede financiar el coste inicial de desarrollo a través de clientes de mayor poder adquisitivo.
- El modelo de negocio de los tokens como objeto de coleccionista ya ha sido probado satisfactoriamente en otros productos, de forma que se reduce el riesgo de que el servicio no acabe ganando escala. En el caso de no desarrollarse más, el resultado seguiría siendo positivo.
  Algunos ejemplos donde se ha utilizado esta idea son en relojes, bolsos y otros accesorios. También en el mercado del arte, vinculando una obra de arte a un token NFT.
- El servicio inicial que se ofrecería sería de uso sencillo. Solamente un token que se puede comprar, vender y visualizar un poco de información sobre éste. De esta forma los usuarios podrían aprender poco a poco a manejarse con una tecnología que todavía desconoce la población general.
- Al trabajar con un número de clientes reducido, se pueden observar y corregir rápidamente los problemas que puedan surgir, ganando experiencia para la posterior incorporación al mercado de masas.

Entre algunos ejemplos en el mercado, dónde se vinculan productos a tokens en la blockchain, nos encontramos a Nike, que está desarrollando unas nuevas zapatillas basadas en la tecnología blockchain de Ethereum. Su nombre es CryptoKicks, las zapatillas se venderán con un archivo digital incluido que permitirá a sus compradores comprobar las propiedades del calzado y verificar su autenticidad. A sus propietarios también se les ofrecerán ventajas exclusivas en ciertos eventos. [45]

También Louis Vuitton ha lanzado una plataforma para verificar la autenticidad de sus productos de lujo, para más adelante ampliar el servicio y avanzar en la trazabilidad de éstos. [46] [47]

No es necesario que la propia marca cree estos tokens, Reebonz es una tienda online que vende productos de lujo (bolsos, joyería, relojes). Cada uno de estos bienes lo vincula con un certificado digital único guardado en la blockchain. Incluye opciones como la “transferencia de certificado” por si quieres revender el producto o la “revocación”, para anularlo por si es robado. [48]

Fase II. Vehículos de alta gama.

Una vez se haya obtenido una respuesta positiva por parte de los clientes durante la Fase I y se haya introducido la tecnología y el funcionamiento más básico de la blockchain a los usuarios, se está en la situación de comenzar la Fase II.

Los objetivos de esta fase son:

- *Ampliar la base de usuarios del servicio de blockchain de forma controlada*.
  Para ello se comenzará a ofrecer el token como certificado de propiedad a los clientes de las marcas de la gama alta. De esta forma se hará llegar la tecnología a un grupo de clientes aún no demasiado amplio, pero ya significativo.
- *Extender los servicios ofrecidos y la usabilidad del token del vehículo*.

Se irán introduciendo poco a poco funciones en el servicio incluido en el token. En este caso sería apropiado introducir el concepto de "pasaporte digital". Así, el usuario iría aprendiendo poco a poco las posibilidades ofrecidas por la blockchain y permitiría a la empresa llevar un seguimiento detallado de la evolución del servicio.
Además, incorporar el pasaporte digital en los modelos de gama alta ofrece un valor muy importante a este tipo de vehículos, ya que son modelos que pierden mucho valor y muy rápido tras la matriculación. Sin embargo, el interés por las gamas premium en el mercado de segunda mano es muy alto, siendo las más buscadas Mercedes, BMW, Mini y Audi respectivamente. [49]
Contar con el pasaporte digital puede aumentar el valor de estos vehículos significativamente en el mercado de segunda mano.

- *Incorporar nuevos agentes en la oferta de servicios*.
  El éxito del servicio de pasaporte digital pasa por que agentes como seguros o talleres participen en el registro de información. En esta fase se pretende demostrarles las ventajas de participar en el uso del servicio de blockchain promoviendo su interés por medio de colaboraciones.
  El objetivo final es que el pasaporte digital se utilice sin la necesidad de que la empresa automovilística influya sobre ellos.

Fase III. Vehículos de todas las gamas.

Una vez se haya conseguido que el token del vehículo sea un elemento conocido por el público y empiece a funcionar de forma autónoma, es decir, sin necesidad de que la empresa que lo emite incentive activamente su utilización, se puede pasar a ampliar al máximo la base de clientes.

En este momento nos encontraríamos en una situación donde los clientes de las gamas de entrada empezarían a reclamar también el token ya que entienden los beneficios que puede aportar y, talleres, seguros y administraciones se encuentran interesados en incorporarlo a sus formas de trabajo.

Es aquí donde se nos ofrece la ocasión de sacarle el máximo partido a la tecnología introduciendo servicios más complejos. En este punto se podría incorporar el servicio de alquiler. Este tipo de actividades, para funcionar de forma exitosa necesitan de una economía de red muy grande, donde muchas personas estén en predisposición de utilizar el servicio. Debido a que ahora muchos de nuestros clientes tendrían en su posesión la llave para poner a prueba el servicio (el token), el servicio de alquiler tendría las mayores probabilidades de resultar en un éxito.

Fase IV. El token como forma de interactuar con el mundo.

Habiéndose ejecutado de forma satisfactoria las fases anteriores del plan estratégico, ahora nos encontramos en un punto en el que todos nuestros clientes tienen un token de propiedad de su vehículo. Estas personas lo utilizan activamente y una gran cantidad de agentes lo han incorporado a su forma de trabajar con los usuarios.

Ahora, ya dentro del terreno de la especulación porque que nos encontramos bastante lejos de la situación de partida, podemos intentar intuir hacia donde se dirigirá una economía basada en tokens.

En esta situación, el token se ha convertido en una forma de interactuar con el consumidor. El token es un excelente mediador entre los clientes y las empresas que quieren ofrecerle un servicio. Sería de esperar que otros emprendedores comiencen a construir formas de ofrecer servicios diferentes, que ahora aún no podemos imaginar, a través de su token de propiedad. También se puede especular que se construyan aplicaciones que hagan interactuar al token de propiedad del vehículo con otros servicios ofrecidos en la blockchain, como otros tokens que sirvan para propósitos diferentes, servicios financieros…

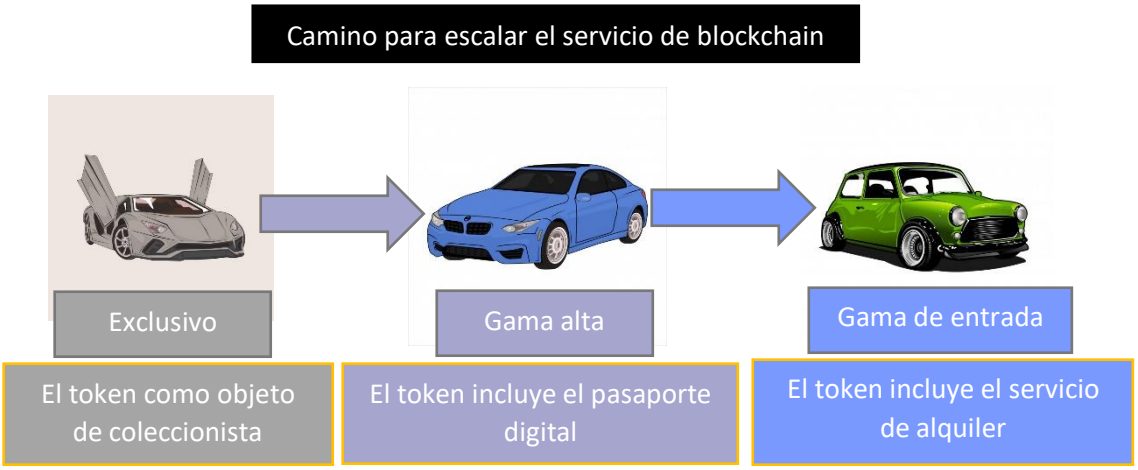

Figura 38. Camino a seguir para conseguir escalar el servicio. Fuente: Elaboración propia.

3.2.2. Consideraciones adicionales

Este modelo presenta algunas virtudes y algunos problemas.

Como se ha visto, uno de los principales factores diferenciales de la tecnología blockchain es su capacidad para no depender de intermediarios que manejen la información y las interacciones de los usuarios.

Esta idea podría entrar en conflicto con el hecho de que una empresa quiera crear un servicio del que en última instancia no posee el control.

La empresa puede posicionarse en un punto intermedio y limitar la "descentralización" de las funcionalidades del token para, dentro de las condiciones de su Smart contract, tomar permisos especiales. Esto no es necesariamente negativo, de hecho, puede tener sus puntos positivos, pero debe ejecutarse con transparencia para evitar la pérdida de confianza en el servicio.

Un ejemplo, sería para solucionar casos de pérdida o hackeo de la cuenta que posee un token. El cliente podría ser víctima de una estafa, hackeo o, sencillamente perder las claves de acceso a su dirección de la blockchain. En circunstancias normales, este es un riesgo que hay que tener en consideración con la tecnología blockchain, sin embargo, la empresa emisora del token podría establecer mecanismos a través de los que, con las pruebas pertinentes, ellos puedan

interceder y recuperarlo. Otra opción sería permitir al usuario decidir si quiere que, para su caso concreto, esta posibilidad exista.

Otros ejemplos de permisos especiales que puede tener la empresa serían, el control de las direcciones que pueden utilizar el Smart contract, permitiendo sólo a ciertos agentes a escribir en él (concesionarios de la marca), o vetando el acceso a otros (si detectasen prácticas abusivas).

Una última estrategia, sería tener un alto control al inicio, y a lo largo del tiempo, en vistas a un desarrollo favorable, irse desprendiendo de estos poderes progresivamente. Existen mecanismos, por medio de la programación, para aplicar cualquiera de las estrategias en el Smart contract que gestiona el token.

La empresa también puede dejar el servicio descentralizado, pero proporcionar interfaces (como aplicaciones, o páginas web propias) para el manejo de las funcionalidades del token. Al ser los primeros y los que mejor conocen su producto, tendrían una amplia ventaja frente a aplicaciones de terceros.

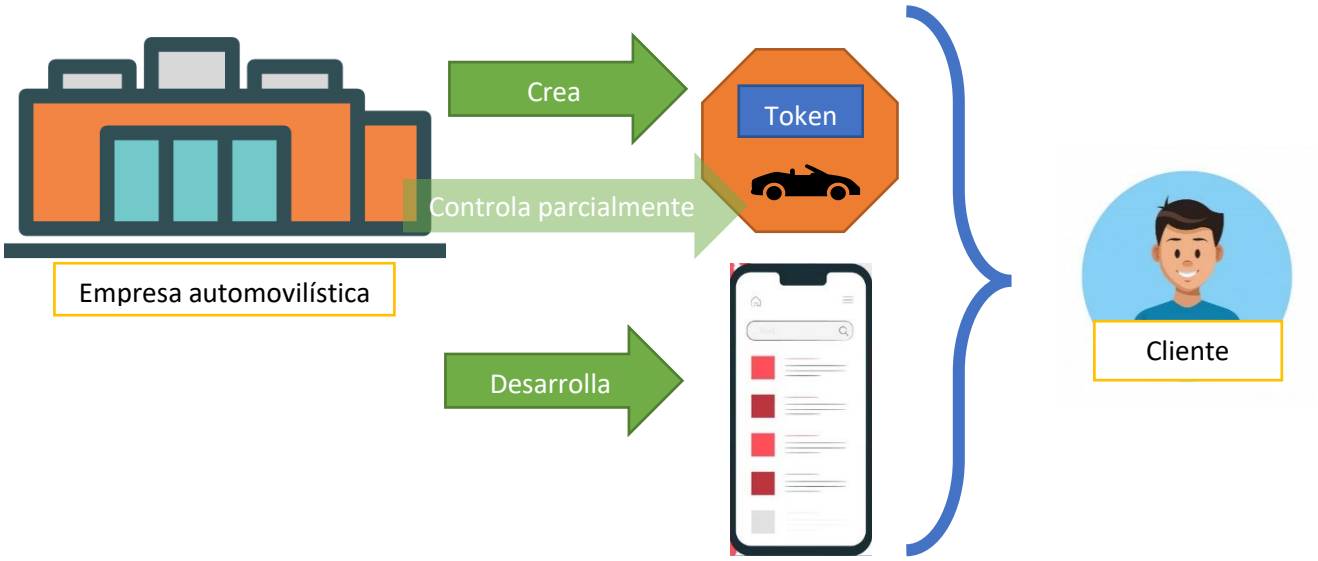

Figura 39. Relación entre la empresa y el token. Fuente: Elaboración propia.

Sin embargo, si lo que se desea es crear un servicio, de nacimiento, descentralizado, se detalla a continuación una posible implementación.

3.3. Implementación independiente

La forma de crear una empresa o un servicio descentralizado en la blockchain es a través de la creación de una Organización Autónoma Descentralizada (DAO).

Una DAO es una organización en la que todas sus actividades están dirigidas a través de unas reglas codificadas en un Smart contract. Entre estas reglas se encuentra el servicio ofrecido y diversos procesos de gobernanza, que generalmente se deciden por votación entre los participantes en la organización.

En esta situación, los participantes y propietarios del servicio serían los accionistas. Las acciones, como es habitual en la blockchain, estarían representadas por un "utility token", una moneda digital que tendría un uso dentro del Smart contract y que comprarían los accionistas esperando que se revalorizase con el éxito del proyecto.

La moneda, al igual que las acciones de una empresa, también otorgará derechos de voto en los procesos de gobernanza del Smart contract. En el caso que compete a este trabajo, estos podrían ser la capacidad de variar ciertos parámetros dentro de las funcionalidades del token y del servicio de postventa ofrecido.

Para que esta moneda tenga valor y además sea creciente si el proyecto gana popularidad, debe establecerse un mecanismo de haga necesaria su demanda para poder utilizar el Smart contract.

A continuación, se va a detallar una forma de llevar a cabo este tipo de implementación. Sin embargo, la programación efectiva de un Smart contract de estas características es bastante compleja y queda fuera del alcance de este trabajo.

3.3.1. Emisión inicial de moneda

Se crean un número de monedas iniciales. Por ejemplo, 100. Las monedas son generadas por la lógica del Smart contract y éste las gestiona, de forma similar a como se hace con el token de propiedad del vehículo. Estas monedas serán para los desarrolladores y financiadores iniciales. Como se ha comentado, las monedas son como acciones de la empresa y se utilizarán para cambiar parámetros dentro del Smart contract. Mientras los desarrolladores dominen la

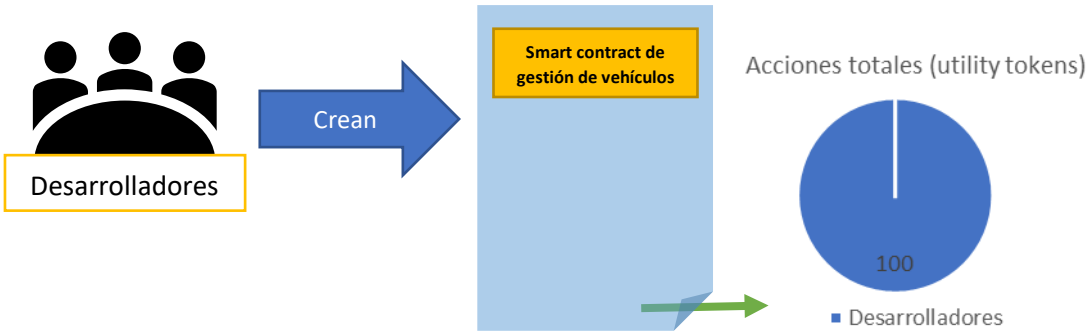

Figura 40. Creación de un smart contract independiente y distribución inicial de sus monedas. Fuente: Elaboración propia.

mayoría de las monedas tendrán control absoluto sobre el servicio, pero es importante que pierdan este dominio a lo largo del tiempo.

3.3.2. Emisión adicional a precio fijo. Inversionistas iniciales

En nuestro modelo de negocio vamos a establecer que, para tener la capacidad de emitir tokens de propiedad del vehículo, se necesita estar en posesión de 10 monedas. Por ello, se buscarán inversores que quieran tener este privilegio en el Smart contract. Diferentes compañías automovilísticas podrían estar interesadas. A cambio de una cantidad de dinero fija, supongamos 100.000$ se emitirían 10 monedas de nueva creación y se entregarían a esta empresa, que podría comenzar a utilizar los servicios descritos en apartados anteriores.

Cuando se hayan conseguido un cierto número de participantes, digamos 10, se dejarán de vender monedas a un precio fijo y se pasará al modelo inflacionario.

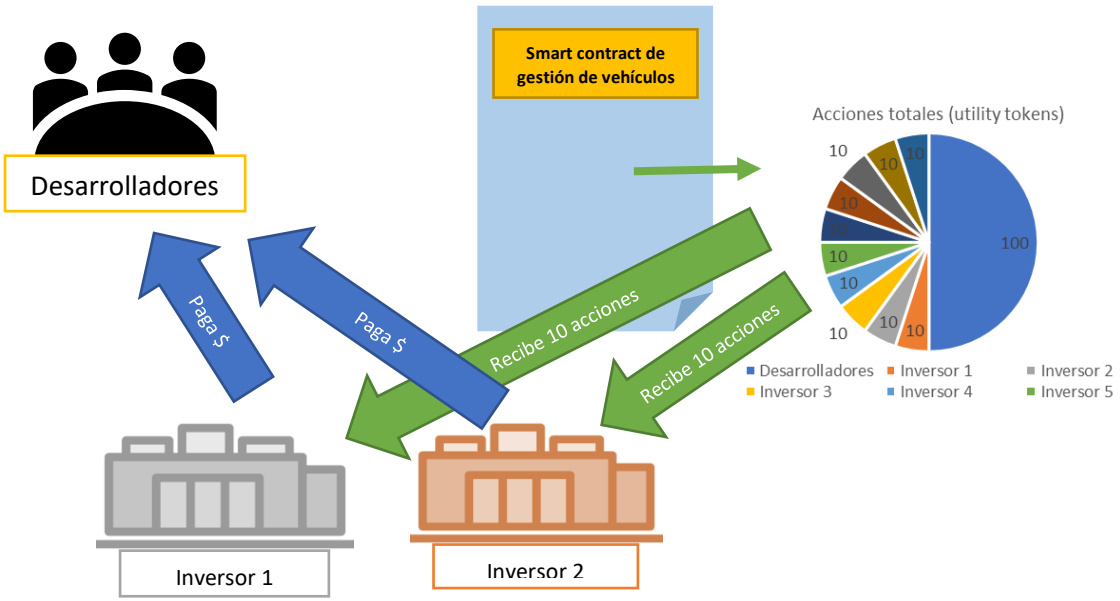

Figura 41. Reparto de monedas a los primeros inversores. Fuente: Elaboración propia.

3.3.3. Modelo inflacionario

Como se ha establecido que la condición para poder emitir tokens de propiedad sea poseer 10 monedas, si el número de monedas es fijo, se limitaría enormemente la cantidad de empresas que puedan acceder a esta función.

Por ello, es necesario que exista una inflación y que se creen nuevas monedas para que se puedan incorporar poco a poco más empresas.

Es importante dar un privilegio especial a los inversionistas iniciales, de forma que no sólo no se vean afectados por la inflación de las monedas que compraron, sino que además se beneficien de haber confiado en el proyecto en una fase temprana.

Por ello, se propone un sistema denominado popularmente como "staking", por el cual los poseedores de moneda reciben un rendimiento anual por tenerlas. De esta forma, los inversores que antes poseían 10 monedas, al final del año podrían poseer 12, si la tasa de inflación se fija en un 20%. Este excedente de 2 monedas se puede vender en el mercado, obteniendo un beneficio y dando la posibilidad a una nueva empresa de conseguir 10 monedas para empezar a emitir sus propios tokens de propiedad.

Por votación entre los poseedores de la moneda, se podría variar esta tasa de inflación en función de las circunstancias.

Figura 42. Situación tras 1 año al 20%. Algunos inversores iniciales han vendido el exceso de moneda en el mercado. Fuente: Elaboración propia.

3.3.4. Modelo deflacionario

También se puede dar la circunstancia de que, llegados a un punto, sea conveniente reducir el número de monedas emitidas. La destrucción de monedas no puede hacerse arbitrariamente, sino que debe seguir la lógica que tendría la recompra de acciones por parte de una empresa normal.

Para ello, se debe encontrar un mecanismo por el cual el Smart contract pueda recuperar monedas emitidas a cambio del valor que ofrece.

Un método apropiado, sería el cobro de una comisión al usuario por el uso del servicio. Supongamos que, al utilizar el servicio de alquiler, del pago se retuviese una comisión (P.ej. del 0,1%). Con este dinero se recompraría en el mercado la moneda que hace el papel de acción y se destruiría.

El valor de esta comisión se podría modificar por votación de los accionistas.

3.3.5. Otras consideraciones

Este modelo de organización consigue que el proyecto nazca de forma descentralizada. El poder para controlar ciertas variables y parámetros, así como su desarrollo futuro, no queda bajo el poder de una empresa concreta, sino que se deja a los accionistas. Los accionistas son principalmente las empresas que participan activamente en él, dándole valor, además de poder ser otros inversores ajenos.

Con el incentivo de aumentar el valor de la acción, los accionistas están interesados en el desarrollo positivo del proyecto, y para los temas que involucren a éste, pasarán de ser competidores a ser colaboradores.

Un entorno de colaboración entre empresas y, la libre entrada de participantes a través de la mediación del Smart contract, favorece la interoperabilidad entre diferentes agentes y la escalabilidad de estos servicios al formar potentes economías de red. Como consecuencia, es esperable que se tienda a formar un ecosistema muy rico y eficiente de servicios en torno al token de propiedad.

Sin embargo, es cierto que la formación de este tipo de círculos virtuosos queda dentro del terreno de la especulación, ya que la tecnología es joven y los negocios que se están empezando a formar en torno a ella, solo acaban de nacer.

4. Costes de la blockchain

Como se expuso en la primera parte de este trabajo, para poder registrar información en la blockchain, es necesario pagar una comisión a los mineros, que son los usuarios que se encargan de generar y validar nuevos bloques, y en última instancia, de ellos depende la seguridad de la red.

Por ello, cada transacción de escritura en la cadena de bloques tiene un coste. El precio está directamente relacionado con la cantidad de información que va a tener que procesar el minero y también depende de la demanda por transacciones que haya en ese momento, ya que pagar una comisión más alta, implica tener prioridad para registrar la transacción.

La comisión se paga en la moneda nativa de la blockchain (en el caso de la blockchain de Ethereum, sería el Ether) cuyo precio depende de la situación del mercado, como si fuese una materia prima.

Podemos hacer un paralelismo con un camión. La carga transportada sería la cantidad de datos que se van a enviar a la blockchain y el Ether la gasolina que consume, que será mayor a más carga y cuyo precio oscila a lo largo del tiempo. Además, si queremos movernos más rápido, gastaremos más gasolina.

Teniendo esto en cuenta, se va a analizar el coste de realizar diferentes acciones en el Smart contract diseñado en este trabajo. Existen diferentes blockchains a disposición de los usuarios, cada una con sus particularidades, para el análisis se tomarán las dos más importantes en el momento de escribir este trabajo: Ethereum y la Binance Smart Chain (BSC).

4.1. Costes en Ethereum

Ethereum es la primera blockchain de Smart contracts que se creó y ha resultado ser muy popular. Sin embargo, debido a los problemas de escalabilidad que presenta actualmente la tecnología y el uso del protocolo de validación Proof of Work, hace que sea cara. Por otra parte, también se la considera la más segura.

a. Costes del despliegue del Smart contract

Subir el Smart contract a la blockchain es un proceso que sólo se debe hacer una vez, pero también es el más costoso, ya que generalmente contiene una gran cantidad de datos que deben ser guardados en la blockchain.

Contrato	Gas	Ether (1 unidad de gas = 29 GWeis)	Euros (1 Ether = 1511 €)
Sólo funciones base de un token NFT	2.230.000	0,06467 Ethers	97,72 €
Base + Función de compraventa	3.020.000	0,08758 Ethers	132,33 €
Base + Función de alquiler	3.260.000	0,09454 Ethers	142,85 €
Base + Función de pasaporte digital	3.560.000	0,10324 Ethers	156,00 €
Smart contract con las 3 funciones.	5.330.000	0,15457 Ethers	233,56 €

Recordamos que el "Gas" es la unidad que mide la cantidad de procesamiento que va a tener que realizar el minero, y el coste es directamente proporcional a su consumo.

En el momento de realizar este trabajo, el precio de una unidad de Gas para conseguir que la transacción sea aceptada en 30 segundos es de 29 GigaWeis (GWei), que recordamos que: $1\ Ether = 1 \cdot 10^9\ GWeis$

A continuación, se muestra el tiempo que tarda en aceptarse una transacción en función del tamaño de la comisión que se paga. El despliegue del Smart contract, al no necesitar ser aceptado al instante, podría realizarse pagando una comisión algo más barata.

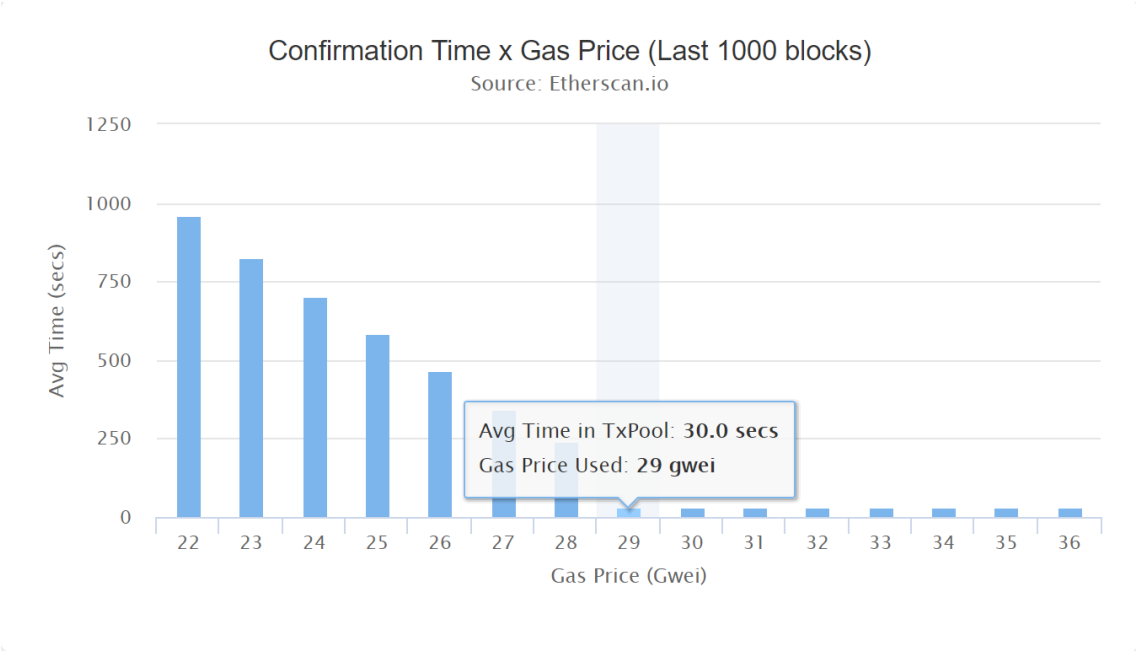

Figura 43. Precio del gas. Fuente: Etherscan

El precio de 1 Ether en el momento de hacer este trabajo es de 1511€. A continuación, se muestra la variación de su precio en el mercado durante el último año.

Figura 44. Gráfico del precio del Ether. Fuente: CoinMarketCap

Un precio de 233 euros para el despliegue del contrato completo puede considerarse bastante económico, teniendo en cuenta el alcance del servicio y que sólo se va a pagar una vez.

b. Costes de interactuar con el Smart contract

En el caso de las operaciones de lectura, se pueden realizar gratuitamente, ya que no producen ningún cambio en la blockchain y, por tanto, no requieren de ningún tipo de procesamiento sobre ésta. Sin embargo, toda acción de compra, venta, generación de nuevos tokens, registro de información en el pasaporte digital… implican la realización de una transacción y la escritura de nuevos datos en la cadena de bloques. Esta escritura requiere de un minero que la incluya y el pago de su correspondiente comisión.

A continuación, se muestra una tabla donde se recoge el coste de utilizar algunas funciones.

Acción	Gas	Ether (en GWeis)	Euros
Emitir un nuevo token	69.300	2.009.700 GWeis	3,04 €
Poner en venta	71.700	2.079.300 GWeis	3,14 €
Comprar	104.300	3.024.700 GWeis	4,57 €
Alquilar	222.900	6.464.100 GWeis	9,77 €
Añadir Revisión	233.900	6.783.100 GWeis	10,25 €
Otorgar permisos de escritura	47.700	1.383.300 GWeis	2,09 €

En el caso de ejecutar funciones del Smart contract, vemos que el coste oscila entre los 2 y 10 euros. Estas cantidades son bastante importantes si se va a hacer un uso continuado de ellas o si el valor aportado es muy pequeño. Para realizar una venta, no importa demasiado, ya que el precio de la comisión es muy inferior al de un coche. Sin embargo, para realizar un pequeño registro en el pasaporte digital, 10 euros pueden suponer demasiado dinero y, para alquilar el vehículo durante un corto periodo de tiempo, la comisión puede suponer una gran proporción del coste total.

Está previsto que la blockchain de Ethereum se actualice con mejoras que buscan reducir sus costes a lo largo de los próximos años, también que surjan competidores a esta red que realicen las mismas funciones a un menor precio. Un competidor actualmente en activo es la Binance Smart Chain.

4.2. Costes en la Binance Smart Chain

La Binance Smart Chain (BSC) es una blockchain de bajo coste que surge para aliviar los problemas de escalabilidad de Ethereum. Su funcionamiento es prácticamente el mismo, lo que cambia es el protocolo de validación, que pasa de Proof of Work (PoW) a Proof of Stake Authority (PSA) (Ver *Otros algoritmos de consenso*). PSA hace que los costes sean mucho menores, pero a costa de limitar mucho la descentralización.

Es por esta razón que la BSC se utiliza habitualmente para delegar operaciones de poco valor desde Ethereum.

La BSC utiliza como moneda la Binance Coin (BNB) que es el equivalente a Ether y su precio de mercado actualmente es de 232 euros. El coste del gas en el caso de esta blockchain es de 5 GWeis. Al funcionar de la misma forma que Ethereum, el consumo de gas en cada transacción es el mismo que antes.

A continuación, se muestra una tabla con los precios de trabajar con la BSC.

Acción	Gas	BNB (1 unidad de gas = 5 GWeis)	Euros (1 BNB = 232 €)
Smart contract con las 3 funciones.	5.330.000	0,02665 BNB	6,18 €
Emitir un nuevo token	69.300	346.500 GWeis	0,08 €
Poner en venta	71.700	358.500 GWeis	0,08 €
Comprar	104.300	521.500 GWeis	0,12 €
Alquilar	222.900	1.114.500 GWeis	0,26 €
Añadir Revisión	233.900	1.114.500 GWeis	0,26 €

En este caso, se observa que el coste de hacer el despliegue de un Smart contract es de unos pocos euros y el de realizar transacciones se queda entre los 8 y los 26 céntimos.

Esto nos demuestra que los costes de utilizar una blockchain pública son viables y que, en los próximos años, a medida que se consiga un correcto equilibrio entre seguridad, descentralización y coste, una gran cantidad de servicios podrán ser acogidos por estas redes.

5. Objetivos de desarrollo sostenible

El avance de la tecnología blockchain y su integración en diferentes apartados de nuestras vidas también ayudará a que se culminen con éxito Objetivos de Desarrollo Sostenible.

De forma directa, servicios como el pasaporte digital ayudan a que la vida de, en este caso, los vehículos de segunda mano, sea más larga, e incentiva a que la conservación de éstos a lo largo de los años sea mejor. Esto ayudará a **reducir los residuos** y la sobreproducción. Del mismo modo, se puede aplicar la idea a otros productos.

La posibilidad de llevar un control de las piezas del vehículo y la posibilidad de relacionarlas con otros servicios de trazabilidad permitirán documentar en la cadena de bloques de forma fiable su procedencia. Así se podrá **impedir** la llegada de **falsificaciones**, incluida la del propio vehículo en sí mismo. Pero extendiendo la idea más en profundidad, se podría trazar la cadena de suministro desde las propias materias primas, **evitando** así aceptar aquellas que procedan de minas donde se **vulneran los derechos humanos**.

Digitalizar la propiedad ayudará también a la creación y consumo de nuevos servicios, que tengan que ver con el manejo de ésta. En este trabajo se propone un servicio de car sharing, donde distintos usuarios pueden compartir sus vehículos, **reduciendo así la superpoblación de coches** en nuestras ciudades. Los servicios relacionados con el manejo de la propiedad digital ayudarán a realizar **actividades empresariales** en todo el mundo, **incluidos países en vías de desarrollo** donde los derechos de propiedad no están igual de regulados que en los países occidentales.

La tokenización de activos y la propiedad digital también ayudarán a la formación de **nuevos mercados**. Un ejemplo, sería la creación de un mercado de energía, donde cada usuario de la red puede aportar, almacenar y consumir energía. Estas actividades estarían mediadas por la tecnología blockchain donde un activo digital se utilizaría como medio para la transmisión de valor. Los **vehículos eléctricos**, un sector de alto crecimiento, serán esenciales para el almacenamiento energético en los hogares, gracias a la tecnología “vehicle to grid” (V2G). Gracias a la propiedad digital, los vehículos eléctricos podrán participar e integrarse en este tipo de mercados a una mayor velocidad y a una gran escala.

6. Conclusiones

6.1. Una nueva forma de entender la propiedad. La propiedad digital.

La tecnología blockchain ha traído al mundo la **propiedad digital**. Por primera vez en la historia el ser humano puede ser dueño de activos de los que tiene total y **completo control**, independientemente de cualquier otra circunstancia. La aproximación más simple a esta idea es la de Bitcoin como "moneda digital", un activo que tiene valor si existe un amplio conjunto de personas que, por sus características como dinero, se lo otorgan. Sin embargo, se puede ir más allá, creando en la blockchain **certificados de propiedad** sobre **activos reales**. Del mismo modo, este certificado tendrá valor si un conjunto de personas lo acepta.

En este segundo caso, al estar relacionadas la propiedad digital y la propiedad real, existen ciertas limitaciones y se requiere de una aceptación sólida y consistente. Esta aceptación se transformará en una realidad tras una fuerte demanda social, atraída por los beneficios que ofrece la tecnología, y la creación de una base legal.

6.2. Integración y desarrollo en crecimiento acelerado.

En la aceptación de este nuevo tipo de activos digitales y la implementación de la tecnología blockchain, han comenzado a trabajar durante los últimos años tanto países como empresas. Recientemente, El Salvador ha aprobado una ley para establecer Bitcoin como **moneda de curso legal** [50], ciudades como Miami estudian permitir el **pago de salarios e impuestos** en esta moneda [51] y otros como el cantón suizo de Zug también aceptan Ether. [52]

En cuanto a los certificados de propiedad, nos encontramos con el Banco de Israel que, durante la redacción de este trabajo, ha informado que se encuentra trabajando en utilizar la tecnología de Ethereum para "programar la **transferencia de un certificado de propiedad** de un automóvil utilizando tokens digitales no fungibles (NFT)". [53]

Empresas como Tesla podrían comenzar a aceptar Bitcoin como pago por sus vehículos [54] y otras, como BMW, se encuentran trabajando en el concepto de "**pasaporte digital**" y la **trazabilidad** de materias primas utilizando la tecnología blockchain de Vechain. [55]

Viendo la dinámica del mercado y la tecnología, se puede intuir que estamos a las puertas de una inminente implantación de las cadenas de bloques en una gran cantidad de ámbitos de nuestro día a día. Entre ellos, la **gestión de datos** por los sistemas de salud y los seguros; la trazabilidad en el transporte y las materias primas; todo tipo de **servicios financieros** ligados al concepto de propiedad digital; negocios de **comunicación** y prensa… e incluso sistemas de **votación** más transparentes y seguros.

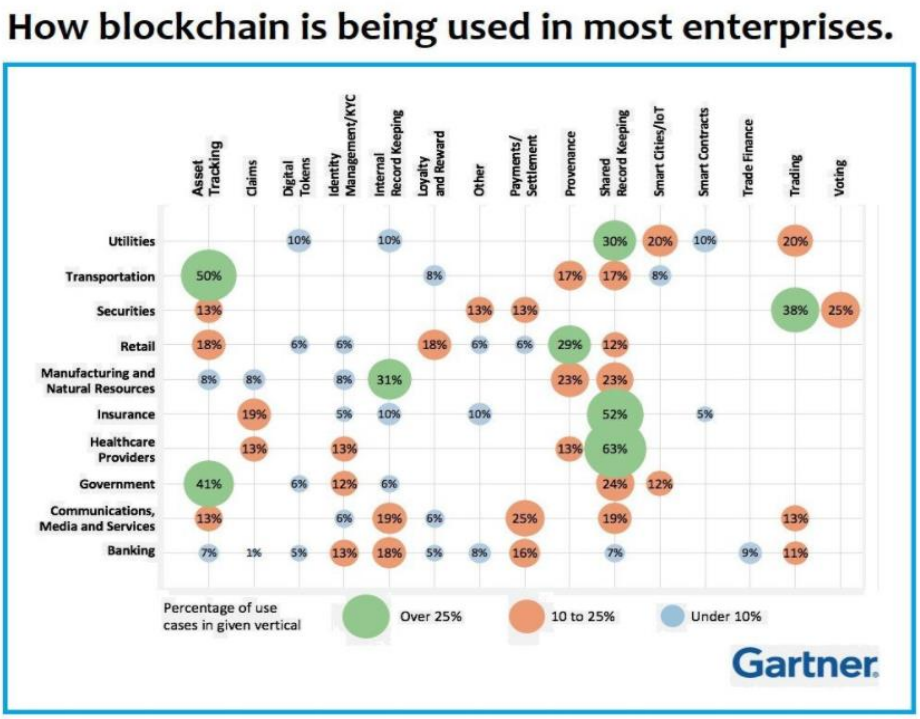

Figura 45. Usos de la blockchain en diferentes industrias. Fuente: Gartner.

Todo ello sin olvidar la llegada de las **Monedas Digitales de los Bancos Centrales** (CBDC), en las que muchos países ya han comenzado a trabajar para crear su propia criptomoneda, siendo China el primero en culminar este proyecto con el yuan Digital.

Figura 46. Estado de desarrollo de las Monedas Digitales del Banco Central en cada país. Fuente: Statista.

6.3. Aplicaciones y servicios en la blockchain.

En este trabajo se propone un modelo para implementar y demostrar el funcionamiento básico del concepto de propiedad digital aplicado al mercado de vehículos. Sobre esta propiedad se construyen unos servicios como es la compraventa, el alquiler y el pasaporte digital.

La idea planteada no debe acabar aquí, sino que se deben visualizar **aplicaciones más extensas** que quedan fuera de los objetivos de este trabajo. Algunos pensamientos ya han sido mencionados a lo largo de la memoria, como relacionar el token del producto (en este caso el coche) con mecanismos de trazabilidad de sus componentes.

El certificado y la **creación de tokens** no deben limitarse a un coche, sino que existe una **gran serie de productos** donde se pueden realizar ejercicios similares. El más cercano a la idea de este trabajo, podría ser la emisión de tokens para maquinaria industrial, que son productos caros, complejos, con muchas piezas de diferentes orígenes y que normalmente se alquilan, por lo que pasan a través de muchas manos de diferentes usuarios.

Por otra parte, siendo conscientes de las diferencias entre unos y otros, pueden ser tokenizados una gran multitud de distintos productos y activos. Casas, terrenos, joyas, ropa…

En el ámbito financiero se podría incluir el certificado digital como contrapartida a la hora de pedir préstamos o dar un aval.

6.4. Limitaciones de escalabilidad.

Para ser completamente funcional y se generalice su uso, la tecnología blockchain debe superar sus limitaciones de escalabilidad. La relativamente escasa cantidad de datos que pueden manejar las actuales cadenas de bloques hace que el coste de usarlas sea demasiado alto para ser viables en determinadas aplicaciones.

Hay que esperar para comprobar si las soluciones propuestas para este problema y actualmente en desarrollo (las llamadas blockchains de tercera generación), consiguen reducir los costes lo suficiente y permiten que la tecnología sea más accesible.

En resumen, la tecnología blockchain permite construir servicios directamente sobre la propiedad, de forma garantista y sin la necesidad de la intermediación de terceros. Esto hace que el acceso a ellos acabe siendo más fácil y libre, tanto para los consumidores como para los oferentes de los servicios.

Tabla de figuras

Anexo I. Smart contract completo.

A continuación, se muestra el código completo, programado en Solidity, propuesto en este trabajo.

Además, el Smart contract se puede encontrar desplegado en la blockchain de la Binance Smart Chain con la dirección: 0x1B4Fd7ad83a113DE3F93E403C308B430597ECd96

Se puede visualizar en el explorador de la blockchain:

https://bscscan.com/address/0x1b4fd7ad83a113de3f93e403c308b430597ecd96

Con ánimos académicos, la función "mint" se encuentra liberada para que todos los usuarios que deseen probar el contrato puedan hacerlo libremente.

También se puede encontrar en la red de pruebas de Ethereum (Ropsten).

https://ropsten.etherscan.io/address/0x1b4fd7ad83a113de3f93e403c308b430597ecd96

Se importa como base del contrato el estándar ERC-721 aquí llamado "NFTVehiculo.sol". Puede encontrarse completo en:

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/ERC721.sol

```
// SPDX-License-Identifier: None

pragma solidity ^0.8.0;

import "./NFTVehiculo.sol";

contract Funcionalidad is NFTVehiculo {


    struct Pagos{
        bool _EstaEnVenta; //Predeterminado en false
        address _comprador;
        uint256 _precio;
        HistVentas[] _HistVentas;
    }

    struct HistVentas{
        address _Propietarios;
        uint256 _ValorDeVenta;
    }

    // mapping de token a Struct de pagos
    mapping (uint256 => Pagos) private _Ventas;
```

```
    address contract_owner;

    constructor(string memory name_, string memory symbol_) NFTVehiculo(n
ame_, symbol_) {
        contract_owner = _msgSender();
    }


// Generar nuevos tokens
    function mint(address to, uint256 tokenId) public {
        require(_msgSender() == contract_owner, "No tienes permisos para
generar tokens");
        _safeMint(to, tokenId);
    }

    function transferOwnership(address to) public{
        require(_msgSender() == contract_owner, "Necesitas ser el propiet
ario para hacer esto");
        contract_owner = to;
    }
```

```
// Aplicación de compraventa
```

```
    function EnVenta(uint256 tokenId, address comprador, uint256 precio)
public returns(address, bool, uint256){
        require(_isApprovedOrOwner(_msgSender(), tokenId), "No eres el pr
opietario o no estas autorizado");     //Si es owner implica que existe _
exists() no necesario
        require(ownerOf(tokenId) != comprador, "No puedes venderte a ti m
ismo");
        _Ventas[tokenId]._comprador = comprador;
        _Ventas[tokenId]._EstaEnVenta = true;
        _Ventas[tokenId]._precio = precio * 1000000000; //En Gweis
        return(_Ventas[tokenId]._comprador, _Ventas[tokenId]._EstaEnVenta
, _Ventas[tokenId]._precio);
    }

    function EstaEnVenta(uint256 tokenId) public view returns(Pagos memor
y) {
        return(_Ventas[tokenId]);
    }


    function _CancelarVenta(uint256 tokenId) internal virtual{
        _Ventas[tokenId]._comprador = address(0);
        _Ventas[tokenId]._EstaEnVenta = false;
        _Ventas[tokenId]._precio = 0;
```

```
    }


    function CancelarVenta(uint256 tokenId) public returns (string memory
) {
        require(_isApprovedOrOwner(_msgSender(), tokenId));
        _CancelarVenta(tokenId);
        return("Ya no esta a la venta");
    }

    function Comprar(uint256 tokenId) public payable {
        require(_Ventas[tokenId]._comprador == _msgSender(), "No estas au
torizado como comprador"); //Solo si has sido designado comprador puede c
omprarlo
        require(msg.value == _Ventas[tokenId]._precio, "El valor de pago
no es correcto");

        address payable owner1 = payable(ownerOf(tokenId));
        uint256 pago = _Ventas[tokenId]._precio;

        _CancelarVenta(tokenId);
        _transfer(owner1, _msgSender(), tokenId);

        owner1.transfer(pago);
        _Ventas[tokenId]._HistVentas.push(HistVentas(_msgSender(), pago))
;       //En el momento de hacer Mint() incluir el primer propietario en
este array

    }

    function VerHistCompradores(uint256 tokenId) public view returns(Hist
Ventas[] memory) {
        return(_Ventas[tokenId]._HistVentas);
    }


// Car sharing

    struct RegistroProp {
            address _direccion;
            uint256 _HoraInic;    //Se podría usar timestamp, Es necesari
o realizar una transaccion para poder cancelar el alquiler de forma efect
iva.
            uint _horas;            //Sería interesante hacer un oráculo
para que se ejecute automáticamente
        }

    struct Alquiler {
```

```
        bool _disponible;           //Marca si el coche esta disponible p
ara poder alquilarse
        bool _alquilado;            //Marca si el coche está alquilado en
 este momento
        uint256 _PrecioHora;
        uint256 _NumUsuarios;       //Num de usuarios que lo han utilizad
o
        RegistroProp[] _RegistroProp;
    }

    mapping (uint256 => Alquiler) private _Alquiler;        //Mapping del
 vehículo a sus datos de alquiler


    struct Registros {
        uint256 _tokenId;
        uint256 _numRegistro;
    }

    struct Historial {
        uint256 _numUsos;
        Registros[] _Registros;
    }

    mapping (address => Historial) private _Historial;          //Mapping
 para guardar la actividad de un usuario

    function PonerEnAlquiler(uint256 tokenId, uint256 PrecioHora) public
{      //Quiza sea necesario un tiempo máximo
        require(_isApprovedOrOwner(_msgSender(), tokenId), "No eres el pr
opietario o no estas autorizado");

        _Alquiler[tokenId]._disponible = true;
        _Alquiler[tokenId]._PrecioHora = PrecioHora * 1000000000; //En GW
eis
    }

    function RetirarDelAlquiler(uint256 tokenId) public {
        require(_isApprovedOrOwner(_msgSender(), tokenId), "No eres el pr
opietario o no estas autorizado");

        _Alquiler[tokenId]._disponible = false;
        _Alquiler[tokenId]._PrecioHora = 0
    }

    function DatosDeAlquiler(uint256 tokenId) public view returns(Alquile
r memory) {
        require(_exists(tokenId), "El Id no existe");
        return(_Alquiler[tokenId]);
```

```
    }

    function _sePuedeAlquilar(uint256 tokenId) internal view virtual retu
rns(bool) {
        return(_Alquiler[tokenId]._disponible);
    }

    function Alquilar(uint256 tokenId, uint256 horas) public payable {
        require(_exists(tokenId), "El Id no existe");
        require(_sePuedeAlquilar(tokenId), "El vehiculo no esta en alquil
er");

        if (_Alquiler[tokenId]._NumUsuarios != 0) {
            require(_FinDelAnterior(tokenId), "El vehiculo sigue en alqui
ler");
        }

        uint256 pagoTotal = _Alquiler[tokenId]._PrecioHora * horas;
        require(msg.value == pagoTotal, "El valor pagado es incorrecto");
        _Alquiler[tokenId]._NumUsuarios += 1;
        uint256 NumUsuarios = _Alquiler[tokenId]._NumUsuarios - 1;  //Par
a que comience con el 0

        _Alquiler[tokenId]._alquilado = true;
        _Alquiler[tokenId]._RegistroProp.push(RegistroProp(_msgSender(),
block.timestamp, horas));

        _GuardarUsuario(tokenId, _msgSender(), NumUsuarios);

        address payable owner1 = payable(ownerOf(tokenId));
        owner1.transfer(pagoTotal);

    }


    function _GuardarUsuario(uint256 tokenId, address user, uint256 numRe
gistro) internal virtual {
        _Historial[user]._numUsos += 1;
        _Historial[user]._Registros.push(Registros(tokenId, numRegistro))
;
    }

    function InfoAlquilerUsuario(address user) public view virtual return
s(Historial memory) {          //Se pasa una address y se obtiene (token,
 num registro)
        return(_Historial[user]);
    }
```

```
    function VerRegistro(uint256 tokenId, uint256 numRegistro) public vie
w virtual returns(RegistroProp memory) {       //Se pasa un (token, num r
egistro) y se obtiene la informacion del alquiler de ese registro en ese
token)
        return(_Alquiler[tokenId]._RegistroProp[numRegistro]);
    }

    function _FinDelAnterior(uint256 tokenId) internal view virtual retur
ns(bool) {         //Nos dice si se ha terminado el tiempo del alquiler a
nterior (En el caso de que no hayamos puesto confirmacion)
        uint256 UltimoUser = _Alquiler[tokenId]._NumUsuarios - 1;
        uint256 HoraInic = _Alquiler[tokenId]._RegistroProp[UltimoUser]._
HoraInic;
        uint256 HoraFin = HoraInic + _Alquiler[tokenId]._RegistroProp[Ult
imoUser]._horas * 1 hours;
        bool HaTerminado;

        if (block.timestamp > HoraFin) {        //Si es la primera vez, b
lock.timestap siempre sera > HoraFin = 0
            HaTerminado = true;
        }
        else {
            HaTerminado = false;
        }
        return (HaTerminado);
    }
```

```
// Pasaporte digital

    //Recambio
    struct RecambioPieza {
        address _agente;        //Agente que realiza la operacion
        uint256 _fecha;         //Fecha de la operacion
        Producto _producto;     //Producto, pieza incorporada
        string _info;           //Informacion adicional
    }

    struct Producto {
        uint256 _productCode;       //Codigo de la pieza
        uint256 _productBatch;      //Número de lote o unidad
        string _productName;        //Nombre de la pieza
        address _productOwner;      //Fabricante
    }

    //Evento
    struct Evento {
        address _agente;
        uint256 _fecha;
```

```
        string _eventName;
        string _info;
    }


    //Revision
    struct Revision {
        address _agente;
        uint256 _fecha;
        uint256 _kilometraje;
        string _resultado;
        string _info;
    }


    //Mappings del tokenId a sus historiales de recambios, eventos y revi
siones
    mapping (uint256 => RecambioPieza[]) _RecambioPieza;
    mapping (uint256 => Evento[]) _Evento;
    mapping (uint256 => Revision[]) _Revision;


    //Mapping aprobados para escribir en el PD
    mapping (address => mapping (uint256 => bool)) private _aprobadoGener
al;        //Mapping de agente => tokenId => true/false
    mapping (uint256 => address) private _permisoUnico;
           //tokenId => direccion agente, cambia a 0 cuando el agente eje
cuta la operacion


    function PermisoUnico(uint256 tokenId, address agente) public{
    //La funcion permite dar permisos de escritura en el PD una vez
        require(_isApprovedOrOwner(_msgSender(), tokenId));
        _permisoUnico[tokenId] = agente;
    }

    function PermisoGeneral(uint256 tokenId, address agente, bool aprobad
o) public{         //La funcion permite dar permisos de forma indefinida
        require(_isApprovedOrOwner(_msgSender(), tokenId));
        _aprobadoGeneral[agente][tokenId] = aprobado;
    }

    function _PuedeEscribir(address agente, uint256 tokenId) internal vir
tual returns(bool) {           //Funcion de ayuda para permisos
        require(_exists(tokenId), "El vehiculo no existe");
        bool puede = (_permisoUnico[tokenId] == agente || _aprobadoGenera
l[agente][tokenId] || _isApprovedOrOwner(agente, tokenId));

        if (_permisoUnico[tokenId] == agente) {
```

```
            _permisoUnico[tokenId] = address(0);
        }

        return(puede);
    }

    function ARecambio(uint256 tokenId, uint256 productCode, uint256 prod
uctBatch, string memory productName, address productOwner, string memory
info) public {
        require(_PuedeEscribir(_msgSender(), tokenId));
        Producto memory _producto1;
        _producto1._productCode = productCode;
        _producto1._productBatch = productBatch;
        _producto1._productName = productName;
        _producto1._productOwner = productOwner;
        _RecambioPieza[tokenId].push(RecambioPieza(_msgSender(), block.ti
mestamp, _producto1, info));
    }

    function VerRecambios(uint256 tokenId) public view virtual returns (R
ecambioPieza[] memory) {
        return (_RecambioPieza[tokenId]);
    }

    function AEvento(uint256 tokenId, string memory eventName, string mem
ory info) public {
        require(_PuedeEscribir(_msgSender(), tokenId));
        _Evento[tokenId].push(Evento(_msgSender(), block.timestamp, event
Name, info));
    }

    function VerEventos(uint256 tokenId) public view virtual returns (Eve
nto[] memory) {
        return(_Evento[tokenId]);
    }

    function ARevision(uint256 tokenId, uint256 kilometraje, string memor
y resultado, string memory info) public {
        require(_PuedeEscribir(_msgSender(), tokenId));
        _Revision[tokenId].push(Revision(_msgSender(), block.timestamp, k
ilometraje, resultado, info));
    }

    function VerRevisiones(uint256 tokenId) public view virtual returns (
Revision[] memory) {
        return(_Revision[tokenId]);
    }
}
```

Anexo II. Procedimiento para interactuar con la blockchain de Ethereum desde una aplicación Android.

A continuación, se muestran los pasos a seguir para poder interactuar con las funciones de lectura y escritura antes programadas en Solidity desde una aplicación de Android utilizando Java.

Se va a utilizar un módulo de web3j

Web3j es una librería que sirve como puente para conectar nuestra aplicación descentralizada con la blockchain de Ethereum.

https://github.com/web3j/web3j

Siguiendo el proceso utilizado en https://medium.datadriveninvestor.com/an-introduction-to-ethereum-development-on-android-using-web3j-and-infura-763940719997

1. *Configurar web3j:*

    a. Para ello se añade web3j al proyecto utilizando un plugin Maven en el archivo *build.gradle: Project*

```
repositories
{
                    mavenCentral()
                    google()
                    jcenter()
                }
```

    b. Ahora se puede añadir web3j como dependencia en el archivo *build.gradle: app*

```
dependencies
{
                implementation fileTree(dir: 'libs', include: ['*.jar'])
                implementation 'com.android.support:appcompat-v7:28.0.0'
                implementation 'com.android.support.constraint:constraint-
            layout:1.1.3'
                testImplementation 'junit:junit:4.12'
                androidTestImplementation 'com.android.support.test:runner:1.0.2'
                androidTestImplementation
            'com.android.support.test.espresso:espresso-core:3.0.2'
                implementation 'org.web3j:core:4.1.0-android'
            }
```

c. También, hay que habilitar permisos de internet en el archivo *AndroidManifest.xml*

```
    <uses-permission android:name="android.permission.INTERNET"/>
```

2. *Conexión a un nodo*

Para conectarse a un nodo de Ethereum es necesario, poseer un nodo o conectarse a un nodo en la nube. Al ser un dispositivo móvil, es muy intensivo en recursos e innecesario ejecutar un nodo propio, por ello se va a realizar una conexión a un nodo proveído por el servicio Infura.

https://infura.io/

La conexión se realiza instanciando un objeto de web3:

```
String node = "https://ropsten.infura.io/v3/----------------------";

Web3j web3 = Web3j.build(new HttpService(node));
```

A partir de aquí, se realiza la interacción con el nodo utilizando la documentación de web3j: https://docs.web3j.io/latest/quickstart/

Una transacción simple, se realizaría de la siguiente forma:

d. Obtención del nonce.

Para evitar duplicidades en el envío de transacciones por una dirección, en Ethereum se lleva un conteo de todas las transacciones realizadas, por ello es necesario comenzar comprobando cuál fue el número de transacción último de la dirección con la que vamos a ejecutar la transacción.

```
EthGetTransactionCount ethGetTransactionCount = web3.ethGetTransactionCount(
        public_address, DefaultBlockParameterName.LATEST).sendAsync().get();

BigInteger nonce = ethGetTransactionCount.getTransactionCount();
```

e. Parámetros de la transacción.

Los parámetros básicos de una transacción en la blockchain de Ethereum son:

i. Nonce: El número de transacción.
ii. Precio del gas: Precio que vamos a pagar por el gas consumido. Si es muy bajo, la transacción puede demorarse mucho o no llegar a ser aceptada.

iii. Límite de gas: Máximo gas que estamos dispuestos a consumir. Si lo superamos la transacción no se completará, pero perderemos el Ether pagado por el gas consumido.
iv. Dirección de destino: Dirección con la que vamos a interactuar.
v. Cantidad de Ether a enviar: La cantidad se debe expresar en weis.

En este caso, no se está interactuando con un contrato inteligente, para ello se añadiría un sexto parámetro "data" con el envío de información. Se explicará más adelante.

```
RawTransaction rawTransaction  = RawTransaction.createEtherTransaction(
        nonce, gasPrice, gasLimit, public_address2, EtherSended);
```

f. Firma del mensaje.

El mensaje ahora debe ser firmado con nuestra clave privada (contenida en "credentials") para poderse verificar la autenticidad de la transacción.

```
byte[] signedMessage = TransactionEncoder.signMessage(rawTransaction,credentials);
```

g. Envío del mensaje.

El mensaje firmado se transforma de formato byte[] a formato hexadecimal. Hecho esto, se envía y se solicita el hash de la transacción para poder realizar un seguimiento de ésta en la blockchain.

```
String hexValue = Numeric.toHexString(signedMessage);

EthSendTransaction ethSendTransaction =
web3.ethSendRawTransaction(hexValue).sendAsync().get();

String transactionHash = ethSendTransaction.getTransactionHash();

Visor_node.setText(transactionHash);
```

3. Interactuar con un contrato inteligente

A la hora de interactuar con un contrato inteligente, es necesario seguir los pasos anteriores de encadenar los parámetros de la transacción, firma y estandarizar el formato de envío. La diferencia es que en esta ocasión hay que añadir un parámetro adicional, que llamaremos "data" en el que se indicará la función del contrato que queremos llamar, el tipo de dato que le vamos a suministrar a la función (ya sea Uint256, Utf8String, address…) y el valor del mismo. Además, si la función devuelve un valor de salida, será necesario especificar el tipo y el valor de éste.

https://github.com/web3j/web3j/blob/master/abi/src/main/java/org/web3j/abi/datatypes/AbiTypes.java

La forma de la función se puede especificar explícitamente (como se va a hacer a continuación) u obtenerse del archivo compilado directamente (ABI) por medio de algún método, si se dispone de éste.

```
Function function = new Function("Nombre_de_la_función",
        Arrays.asList(new Uint256(123),
                      new Utf8String("Texto"),
                      new Uint32(1230),
                      new Uint32(1230)),  // Parameters to pass as Solidity Types
        Collections.emptyList());  //Si no hay valor de salida.
       // Si hubiese valores de salida se indicarían de la forma:
       // Arrays.asList(new TypeReference<Type>() {}, ...)) //Siendo Type el tipo de parámetro
```

Solidity nombra internamente a las funciones de un contrato de la siguiente manera:

- Toma el nombre de la función con la lista, entre paréntesis, de los tipos de parámetros (sin el valor), separados por comas y sin espacios. Esta es la firma de la función.
- Realiza el hash (Keccak-256) de la firma.
- Toma los cuatro primeros bytes del hash.
- El resultado quedaría de la siguiente forma: 0x17b4bc1e. A este valor se le denomina *selector* de la función.

Esta es la razón por la que se debe definir la función de la forma en que se ha hecho.

A continuación, se utiliza un método que codifica los datos, de forma que calcula el selector y anexa los valores de input.

```
String encodedFunction = FunctionEncoder.encode(function);
```

Ahora, de la misma forma que se hacía en una transacción simple, se definen los parámetros de nonce, precio del gas, límite de gas, dirección del contrato (antes dirección del destinatario), ether a transferir (expresado en weis) (que será cero si no se requiere transferir ether) y en esta ocasión, también la función codificada.

```
RawTransaction rawTransaction = RawTransaction.createTransaction(nonce, gasPrice, gasLimit, contractAddress, Ether, encodedFunction);
```

Es importante controlar el límite de gas, ya que puede variar mucho dependiendo de la función que se va a llamar y el contrato. Recordemos que, si el límite es inferior al coste de procesamiento, la operación se revertirá.

Finalmente se firman y se envían los datos de la transacción.

```
byte[] signedMessage = TransactionEncoder.signMessage(rawTransaction, credentials);
String hexValue = Numeric.toHexString(signedMessage);

EthSendTransaction ethSendTransaction =
web3.ethSendRawTransaction(hexValue).sendAsync().get();
```

4. Leer datos de un contrato inteligente

Ethereum es una blockchain pública y se pueden leer libremente los datos contenidos en ella. Para una lectura legible de la información (más allá de leer los bytes en bruto del contrato) se necesita conocer el *selector* de la función de lectura del parámetro que queremos conocer. Esto se consigue de la misma forma que se hacía en el punto anterior. También se pueden leer de esta forma variables de tipo "public".

A diferencia de las funciones de escritura, donde se envía información al contrato para que se ejecuten una serie de procesos, las funciones de lectura no modifican el contrato y, por tanto, tampoco requieren de procesamiento. Por esta razón no se consume gas al realizar la operación ni se requiere de una clave privada.

A continuación, se describe el proceso de lectura de información:

En primer lugar, se define la función que se quiere leer. En este caso, la lista de los parámetros de entrada estará vacía.

```
Function function = new Function(
        "Nombre_de_la_función",
        Collections.emptyList(),
        Arrays.asList(new TypeReference<Utf8String>() {},
                new TypeReference<Uint32>() {},
                new TypeReference<Uint32>() {}
                ));

String encodedFunction = FunctionEncoder.encode(function);
```

En segundo lugar, se genera la llamada entregando como parámetros, la dirección pública desde la que se emite la llamada, el contrato que se va a leer y la función codificada que se quiere leer.

```
Transaction call = Transaction.createEthCallTransaction(public_address,
contractAddress, encodedFunction);
EthCall response =
web3.ethCall(call,DefaultBlockParameterName.LATEST).sendAsync().get();
```

Por último, se crea una lista que reciba la información.

```
List<Type> someTypes = FunctionReturnDecoder.decode(
        response.getValue(), function.getOutputParameters());
```

Bibliografía

[1] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Cadena_de_bloques.

[2] «Microsoft,» [En línea]. Available: https://docs.microsoft.com/en-us/archive/msdn-magazine/2018/july/blockchain-decentralized-applications-with-azure-blockchain-as-a-service.

[3] S. Nakamoto, «Bitcoin.org,» [En línea]. Available: https://bitcoin.org/files/bitcoin-paper/bitcoin_es.pdf.

[4] [En línea]. Available: https://link.springer.com/content/pdf/10.1007/BF00196791.pdf.

[5] [En línea]. Available: https://blog.addalia.com/historia-del-blockchain.

[6] [En línea]. Available: https://www.welivesecurity.com/la-es/2018/09/04/blockchain-que-es-como-funciona-y-como-se-esta-usando-en-el-mercado/.

[7] C. Dolader Retamal, J. Bel Roig y J. L. Muñoz Tapia. [En línea]. Available: https://www.mincotur.gob.es/Publicaciones/Publicacionesperiodicas/EconomiaIndustrial/RevistaEconomiaIndustrial/405/DOLADER,%20BEL%20Y%20MU%C3%91OZ.pdf.

[8] «Tradeix,» [En línea]. Available: https://tradeix.com/distributed-ledger-technology/.

[9] [En línea]. Available: https://hackernoon.com/gaining-clarity-on-key-terminology-bitcoin-versus-blockchain-versus-distributed-ledger-technology-7b43978a64f2.

[10] «Medium,» [En línea]. Available: https://medium.com/@akadiyala/nuances-between-permissionless-and-permissioned-blockchains-f5b566f5d483.

[11] J. Barrios. [En línea]. Available: https://www.juanbarrios.com/la-tecnologia-blockchain-continua-fortaleciendose/.

[12] «Bit2me,» [En línea]. Available: https://academy.bit2me.com/como-funciona-el-hash-en-bitcoin/.

[13] «Coursera,» [En línea]. Available: https://www.coursera.org/learn/blockchain-business-models/lecture/tBkem/brute-forcing-a-sha-256.

[14] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Firma_digital.

[15] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Criptograf%C3%ADa_de_curva_el%C3%ADptica.

[16] [En línea]. Available: https://actualidadcripto.com/llaves-de-bitcoin-y-curva-eliptica-secp256k1/.

[17] «Coursera,» [En línea]. Available: https://www.coursera.org/learn/blockchain-business-models/lecture/S2lc9/ecdsa.

[18] [En línea]. Available: https://www.bitcobie.com/protocolos-de-consenso-pilar-en-la-seguridad-blockchain/.

[19] «Binance,» [En línea]. Available: https://academy.binance.com/es/articles/what-is-a-blockchain-consensus-algorithm.

[20] «Bit2me,» [En línea]. Available: https://academy.bit2me.com/mineria-bitcoin-como-se-crea-un-bloque/.

[21] «Bitcoin.it,» [En línea]. Available: https://en.bitcoin.it/wiki/Block_hashing_algorithm.

[22] A. Butler. [En línea]. Available: https://hackernoon.com/ethereum-classic-attacked-how-does-the-51-attack-occur-a5f3fa5d852e.

[23] «Coursera,» [En línea]. Available: https://www.coursera.org/learn/blockchain-business-models/lecture/mEM76/proof-of-work.

[24] «Btc.com,» [En línea]. Available: https://btc.com/stats/pool.

[25] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Prueba_de_participaci%C3%B3n.

[26] [En línea]. Available: https://academy.bit2me.com/transacciones-bitcoin/.

[27] «Bit2me,» [En línea]. Available: https://academy.bit2me.com/como-se-crea-una-direccion-bitcoin.

[28] [En línea]. Available: https://www.foxize.com/blog/bitcoin-pros-contras-moneda-internet/.

[29] «Bit2me,» [En línea]. Available: https://academy.bit2me.com/que-son-los-smart-contracts/.

[30] «Cointelegraph,» [En línea]. Available: https://es.cointelegraph.com/explained/learn-how-to-install-an-ethereum-node-using-geth-and-docker.

[31] «El confidencial,» [En línea]. Available: https://blogs.elconfidencial.com/economia/laissez-faire/2019-08-26/bitcoin-alternativa-dolar_2191991/.

[32] «Bit2me,» [En línea]. Available: https://academy.bit2me.com/que-es-utility-token/.

[33] «Kraken,» [En línea]. Available: https://www.kraken.com/es-es/learn/what-is-aave-lend.

[34] «Bit2me,» [En línea]. Available: https://academy.bit2me.com/que-es-un-security-token/.

[35] «Bit2me,» [En línea]. Available: https://academy.bit2me.com/que-es-un-security-token/.

[36] M. Thoma. [En línea]. Available: https://ichi.pro/es/los-7-tipos-de-criptomonedas-que-debes-conocer-65037394644664.

[37] «Binance,» [En línea]. Available: https://academy.binance.com/es/articles/blockchain-oracles-explained.

[38] «CNN,» [En línea]. Available: https://cnnespanol.cnn.com/2021/06/22/bitcoin-desplome-china-criptomonedas-trax/.

[39] [En línea]. Available: https://invezz.com/es/noticias/2021/06/07/vechain-vet-podria-asociarse-con-el-gobierno-chino-para-expandir-blockchain/.

[40] [En línea]. Available: https://www.ecologistasenaccion.org/wp-content/uploads/adjuntos-spip/pdf/info_cuentas-ecologicas.pdf.

[41] «Medium,» [En línea]. Available: https://l4mp1.medium.com/blockchain-based-car-sharing-e71c40754c1f.

[42] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Uso_temporal_de_veh%C3%ADculos.

[43] «nhtsa,» [En línea]. Available: https://www.nhtsa.gov/es/equipo/fraude-de-odometro.

[44] [En línea]. Available: https://www.autocasion.com/actualidad/noticias/las-itv-registraran-el-kilometraje-del-vehiculo-desde-el-proximo-ano.

[45] [En línea]. Available: https://www.blockchainservices.es/novedades/cryptokicks-las-zapatillas-de-nike-basadas-en-blockchain/.

[46] [En línea]. Available: https://www.modaes.es/empresa/lvmh-avanza-en-blockchain-y-lanza-una-plataforma-para-verificar-la-autenticidad-de-sus-productos.html.

[47] [En línea]. Available: https://www.modaes.es/empresa/lvmh-prada-y-richemont-arrancan-un-proyecto-de-blockchain-para-avanzar-en-trazabilidad.html.

[48] [En línea]. Available: https://www.bloomberg.com/press-releases/2020-01-23/reebonz-implements-blockchain-solution-to-digitize-products-becomes-first-fashion-technology-company-to-provide-end-to-end.

[49] [En línea]. Available: https://noticias.coches.com/informes/marcas-y-modelos-de-coches-mas-buscados-segunda-mano/344560.

[50] «La razón,» [En línea]. Available: https://www.larazon.es/internacional/20210619/rtoab2aeibcx5gm4lj2yxxborq.html.

[51] «Xataka,» [En línea]. Available: https://www.xataka.com/criptomonedas/dolares-tambien-bitcoin-miami-estudian-pagar-salarios-permitir-pagar-impuestos-criptomoneda.

[52] «Cointelegraph,» [En línea]. Available: https://es.cointelegraph.com/news/swiss-canton-of-zug-starts-accepting-tax-payments-in-cryptocurrency.

[53] «Cointelegraph,» [En línea]. Available: https://es.cointelegraph.com/news/bank-of-israel-steps-up-cbdc-efforts-with-reported-tests-on-ethereum..

[54] «El pais,» [En línea]. Available: https://cincodias.elpais.com/cincodias/2021/07/22/mercados/1626954992_738608.html.

[55] «BMW,» [En línea]. Available: https://www.bmw.com/es/innovation/tecnologia-blockchain.html.

[56] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Funci%C3%B3n_hash.