URL SHA256 => <https://academy.bit2me.com/wp-content/uploads/2019/10/Criptography_SHA_256_es.pdf>

URL SHA256 => <https://sha256algorithm.com/>

URL: <https://101blockchains.com/es/historia-de-la-blockchain/>

URL: <https://academy.binance.com/es/articles/history-of-blockchain>

URL: <https://www.anf.es/pdf/Haber_Stornetta.pdf>

URL: https://bitcoin.org/bitcoin.pdf

1. ¿Qué es el blockchain?

**¿Quién inventó las cadenas de bloques?**

El concepto de blockchain surge en 1991 tras la redacción de un documento denominado “*How to Time-Stamp a Digital Document*” redactado por **Stuart Haber** y **W. Scott Stornetta**.

“*La perspectiva de un mundo en el que todos los documentos de texto, audio, imagen y vídeo estén en digital en plataformas fácilmente modificables plantea la cuestión de cómo certificar cuándo se creó o modificó por última vez un documento. El problema es cómo podemos poner un sello de tiempo a los datos, no al soporte. En este informe proponemos procedimientos computacionalmente prácticos para el sellado de tiempo digital de tales documentos, de modo que sea inviable que un usuario retrase o adelante la fecha de su documento, incluso con la confabulación de un servicio de sellado de tiempo. Nuestros procedimientos mantienen la total privacidad de los de los documentos y no requieren ningún registro por parte del servicio de sellado de tiempo*.”

Como vemos en la introducción de este informe Haber y Stornetta nos van a proponer soluciones para que un archivo que recorre internet pueda tener un sello *timestamp* inmodificable por el autor del documento. También nos hablan de la importancia que podría tener esto en campos como la creación de patentes evitando conflictos por cual pudo generarse antes.

Nos informan que el método de sellado de documentos digitales necesita de dos propiedades básicas:

* En primer lugar, hay que encontrar una manera de marcar el tiempo, sin depender de las características del soporte en el que aparecen los datos, de modo que sea imposible cambiar incluso un bit del documento sin que el cambio sea evidente
* En segundo lugar, debería ser imposible estampar un documento con una fecha y hora diferentes a las reales.

La propuesta que nos hacen ambos ingenieros en dicho documento es la siguiente:

En el caso de que queramos realizar un sellado de tiempo a nuestro documento, deberemos pasarlo por una función la cual nos devolverá un hash. Esta función debe ser muy sensible ya que con sólo cambiar un bit del documento el hash deberá ser totalmente distinto. Una vez tenemos nuestro documento con la información correspondiente y su hash lo enviaremos a un servicio de sellado de tiempo. Este servicio firmará con su firma digital nuestro documento y nos lo devolverá. De esta manera los documentos serán sellados temporalmente y no podrán ser modificados ya que obtendríamos un hash totalmente diferente.

Sin embargo, no es hasta 2008 tras el lanzamiento del primer informe redactado bajo el pseudónimo de **Satoshi Nakamoto** cuando la tecnología del blockchain empieza obtener relevancia.

Este informe aporta ideas para transaccionar dinero sin necesitar terceras entidades. Aunque la idea de la firma digital de Haber y Stornetta es una parte de la solución, para Nakamoto cree que perdemos todos los beneficios al necesitar a una tercera entidad en la que confiar nuestros datos y evitando el doble de gastos.

Es por ello que Nakamoto propone la idea de introducir una red *peer-to-peer****.*** Esta red tendrá que disponer de las siguientes características:

* Las nuevas transacciones se difunden a todos los nodos.
* Cada nodo recopila las nuevas transacciones en un bloque.
* Cada nodo trabaja para encontrar una prueba de trabajo difícil para su bloque.
* Cuando un nodo encuentra una prueba de trabajo, difunde el bloque a todos los nodos.
* Los nodos aceptan el bloque sólo si todas las transacciones que contiene son válidas.
* Los nodos expresan su aceptación del bloque trabajando en la creación del siguiente bloque en la cadena, utilizando el hash del bloque aceptado como hash anterior.

Como vemos Nakamoto nombra el concepto de *prueba de trabajo* introducido en 2004 por el informático Hal Finney. La prueba de trabajo consiste en lo siguiente:

Una vez hemos obtenido el hash de nuestro bloque mediante una función de hash, como podría ser **sha-256,** introduciremos una nueva variable para que ese bloque sea añadido a nuestra cadena de bloques. El hash de nuestro bloque deberá empezar por una cifra de ceros y mediante la prueba de trabajo deberemos adivinar el número, **nonce,** que junto con la información del bloque y el hash previo del bloque anterior conforman el hash del bloque que queremos añadir a nuestra cadena.

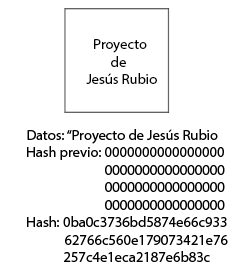
Una vez se descubra el nonce, los otros nodos deberán comprobar que efectivamente es el número correcto para formar el hash del bloque. Una vez los nodos estén mayoritariamente de acuerdo el bloque será añadido a la cadena.

**¿Qué es un bloque?**

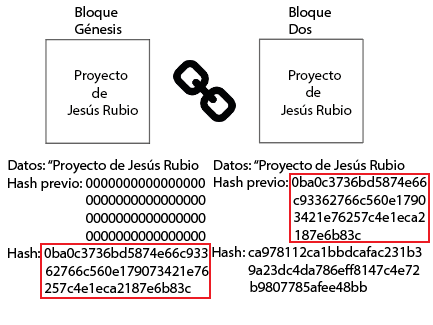
Un bloque es un registro el cual contiene datos en su interior. A dicho bloque, le vamos a añadir el hash, término que explicaremos más en profundidad, del **bloque anterior.** De esta manera todos los bloques estarán enlazados entre sí. Por lo tanto, contamos con la información que contiene el bloque y el hash del bloque anterior, a partir de estas variables podremos generar el hash, el identificador, de nuestro bloque.

Vamos a explicar este concepto para que quede mucho más claro a través de representación gráfica.

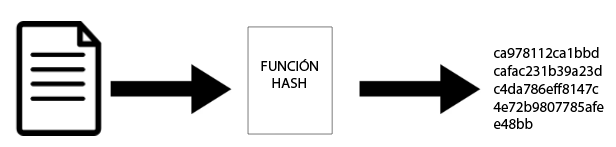
Lo primero que debe tener una cadena de bloque es el **bloque génesis** o el **bloque generador.** Este será el bloque a partir del cual se generarán los siguientes, este bloque **siempre** será el primer bloque. La principal característica de este bloque es su hash previo, ya que al ser el primero estará formado por todo ceros. A modo de ejemplo nuestro bloque generador constará de un texto con información, su hash previo y su hash propio.



A partir de nuestro bloque génesis llegarán los siguientes. Siempre guardando el hash del bloque previo, es decir, están **enlazados criptográficamente**. Si ahora añadiéramos un segundo bloque a nuestra cadena de ejemplo, este bloque debería conservar íntegramente el hash del bloque inminentemente anterior, en este caso el bloque génesis.

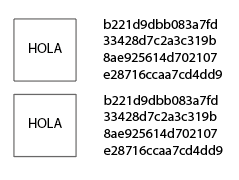
**El hash**

Un hash es un número hexadecimal de 64 dígitos con el cuál podemos identificar cualquier objeto digital (documento, video, foto, audio, etc). Este número es la salida que proporciona una función hash al nosotros introducirle como parámetro de entrada el objeto que queremos codificar, y eso es lo que queremos hacer codificar, no queremos comprimir o empaquetar nuestro documento en 64 dígitos, queremos obtener un identificador para nuestro documento.

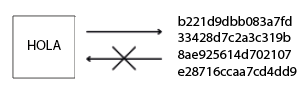


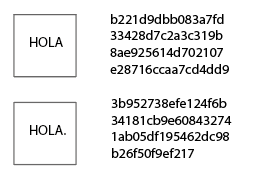
Propiedades deseables en una función hash.

* Permite ejecutarse sobre contenido digital de cualquier tamaño y formato. Es conveniente que podamos obtener un hash sin importar el tipo de objeto que queramos hashear.
* Determinista, es decir, para el mismo mensaje o conjunto de datos de entrada siempre se obtiene el mismo resultado.



* Unidireccional, reconstruir el mensaje original a partir del resultado de la función hash debe ser extremadamente costoso, sino imposible.



* Efecto avalancha, la mínima variación en el mensaje original ha de producir un hash totalmente distinto.

* Cómputo elevado, la calculación del hash debe ser casi instantánea sin importar el peso del documento de entrada.
* Soporte de colisiones, puede ser que ficheros con diferente contenido puedan tener el mismo hash. Aunque el número de hashes que disponemos es gigantesco el número de ficheros distintos es aún más grande por lo que puede darse el caso de que ficheros tengan el mismo hash.

Una de las funciones de hash más usadas actualmente es la conocida **SHA-256** diseñadas por la Agencia de Seguridad Nacional (NSA) en 2001. Como ya hemos mencionado anteriormente el objetivo de esta función es resumir cualquier objeto digital en 256 bits de longitud expresados en un número hexadecimal de 64 caracteres denominado hash.

El procedimiento que realiza esta función hash cuando recibe un objeto de entrada es el siguiente:

Lo primero que va a realizar es pasar el mensaje de entrada a hexadecimal. Para ello mirará el valor de cada carácter en la tabla ASCII y será ese número el que pasará a hexadecimal.

Carácter => ASCII => Hexadecimal

J 74 4A

e 101 65

s 115 73

ú 163 A3

s 115 73

Ahora debemos unir los valores en hexadecimal de nuestro mensaje:

Jesús => Hexadecimal => 4A6573A373

Blockchain => Hexadecimal => 426C6F636B636861696E

Gonzalo Nazareno => Hexadecimal => 476F6E7A616C6F204E617A6172656E6A

Nótese que los espacios en blanco también ocupan un valor en la tabla ASCII, en concreto el valor 32.

El valor en hexadecimal de nuestro mensaje será guardado en una variable que será usada posteriormente por otros procedimientos.

A continuación, calculamos la longitud de nuestro mensaje en bits multiplicando por 8 bits, el valor de cada carácter ASCII. También almacenaremos este valor en otra variable.

Mensaje => Bits x Nº de caracteres => Longitud en Hexadecimal

Jesús 8 x 5 = 40 28

Blockchain 8 x 10 = 80 50

Gonzalo Nazareno 8 x 16 = 128 80

Una vez tenemos el mensaje y la longitud de este el algoritmo pasará a construir la siguiente variable, la cual se trata de un array, denominada W, de 64 elementos que contiene palabras hexadecimales de 32 bits. Esta variable será inicializada por todo ceros y construida por intervalos, siendo el primer intervalo los 16 primeros registros y el segundo intervalo los 48 restantes, será construida a partir de la siguiente fórmula que explicaremos a continuación.

Trozo *n* del mensaje si 0 ≤ n < 16

Array W

σ1(Wn-2) + Wn-7 + σ0(Wn-15) + Wn-16  si 16 ≤ n < 64

En el primer intervalo se reservarán los 16 primeros registros para almacenar nuestro mensaje de entrada en formato hexadecimal en separaciones de 32 bits:

32 bits 32 bits

Jesús => Hexadecimal => 4A6573A 373

32 bits 32 bits 32 bits

Blockchain => Hexadecimal => 426C6F63 6B636861 696E

32 bits 32 bits 32 bits 32 bits

Gonzalo Nazareno => Hexadecimal => 476F6E7A 616C6F20 4E617A61 72656E6A

Como es lógico no todos los bloques ocuparán los 32 bits exactos, es por ello que el algoritmo los dejará vacíos y se encargará de rellenarlos en procesos posteriores. Por lo tanto, ya tenemos nuestro primer intervalo con un tamaño de 512 bits dividido en 16 bloques de 32 bits.

Ahora tomará un bit que represente el número 1 y lo desplazará al bit más a la izquierda de un byte, por último, lo convertirá en hexadecimal:

1 => binario => 00000001 => pasamos el 1 al bit más a la izquierda => 10000000 => hexadecimal => 80

Añadimos el valor que acabamos de calcular a la derecha de nuestro mensaje:

Jesús => 4A6573A + 373 + 80

Blockchain => 426C6F63 + 6B636861 + 696E + 80

Gonzalo Nazareno => 476F6E7A + 616C6F20 + 4E617A61 + 72656E6A + 80

A continuación, se añaden 0 hasta llegar a los 448 bits y se cierra con dos bloques de 32 bits con la longitud de nuestro mensaje. En caso de que la longitud no ocupe los 32 bits rellenaremos con 0 por la izquierda.

4A6573A + 373 + 80 + 000 + 00000000 + 00000000 + 00000000 + 00000000 + Jesús 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000028

4A6573A + 6B636861+ 696E + 80 +00 + 00000000 + 00000000 + Blockchain 00000000+ 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000050

476F6E7A + 616C6F20 + 4E617A61 + 72656E6A + 80 + 000000 +

Gonzalo Nazareno 00000000+ 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000000 + 00000000 +

00000080

El algoritmo ahora inicializará dos variables, la constante K formada por 64 palabras hexadecimales y las 8 variables h que igualmente se compone de 8 palabras hexadecimales.

La constante K queda formada por 64 palabras hexadecimales compuestas por los primeros 32 bits en hexadecimal del resultado de la parte fraccionaria, decimales, de la raíz cúbica de cada uno de los primeros 64 números primos.

2 = 1.2599210498948732 => Parte fraccionaria => 0.2599210498948732 => Hexadecimal => 0.428A2F98D728A242D2B4 => 32 bits => 428A2F98

3 = 1.4422495703074083=> Parte fraccionaria => 0.4422495703074083 => Hexadecimal => 0.7137449123EF5FDF4D8D => 32 bits => 71374491

Esta operación será realizada con los 62 siguientes números primos.

De igual forma calculará las 8 palabras h pero esta vez serán calculadas a través de los 32 primeros bits de la parte fraccionaria en hexadecimal de la raíz cuadrada de los 8 primeros números bits. Estas 8 palabras serán correlacionadas con las variables A, B, C, D, E, F, G y H respectivamente.

A = 6A09E667

B = BB67AE85

C = 3C6EF372

D = A54F F53A

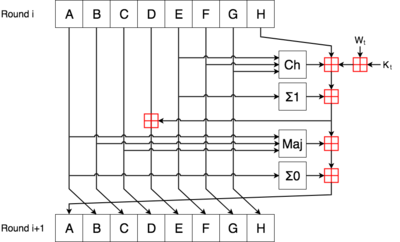
E = 510E527F

F = 9B05688C

G = 1F83D9AB

H = 5BE0CD19

Una vez el algoritmo tiene todos los ingredientes podrá empezar a encriptar nuestro mensaje.



Lo primero que debemos de tener en cuenta es que las variables A, B, C, D, E, F, G y H siempre tendrán el mismo valor en la primera ronda, será a partir de la siguiente cuando empezarán a tomar valores totalmente diferente siguiendo las operaciones que vamos a explicar a continuación.

**Variable A**

* El valor de A en cada ronda será el que tenga B justo en la siguiente**.**
* El valor de A en cada ronda será pasado a la función M para obtener el valor de A en la siguiente ronda.
* Por último, también se pasará su valor a la función para obtener junto con la función M el valor de A en la siguiente ronda.

**Variable B**

* El valor de B en cada ronda será el que tenga C justo en la siguiente**.**
* El valor de B en cada ronda será pasado a la función M para obtener el valor de A en la siguiente ronda.

**Variable C**

* El valor de C en cada ronda será el que tenga D justo en la siguiente**.**
* El valor de C en cada ronda será pasado a la función M para obtener el valor de A en la siguiente ronda.

**Variable D**

* Será uno más de los elementos junto con Wi, Ki, la variable H y E, las funciones y Ch. Con la suma en módulo 32 de estos elementos calcularemos el valor de E en la siguiente ronda.

**Variable E**

* El valor de E en cada ronda será el que tenga F justo en la siguiente**.**
* El valor de E en cada ronda será pasado a la función y Ch para obtener el valor de A en la siguiente ronda.

**Variable F**

* El valor de F en cada ronda será el que tenga G justo en la siguiente**.**
* El valor de F en cada ronda será pasado a la función Ch para obtener el valor de A en la siguiente ronda.

**Variable G**

* El valor de G en cada ronda será el que tenga H justo en la siguiente**.**
* El valor de G en cada ronda será pasado a la función Ch para obtener el valor de A en la siguiente ronda.

**Variable H**

* Será uno más de los elementos junto con Wi, Ki, la variable D y E, las funciones y Ch. Con la suma en módulo 32 de estos elementos calcularemos el valor de E en la siguiente ronda.

**Función Ch**

Es una función que realizará operaciones lógicas tomando como parámetros de entrada las variables E, F y G. Su fórmula es la siguiente:

Ch (E,F,G) = (E F) (¬E G)

**Función Maj**

Es una función que realizará operaciones lógicas tomando como parámetros de entrada las variables A, B y C. Su fórmula es la siguiente:

Maj (A,B,C) = (A B) (A C) (B C)

**Función**

Es una función que realizará operaciones lógicas y operaciones binarias rotativas tomando como parámetros de entrada la variable Ai. Su fórmula es la siguiente:

(x) = ROT R2 (x) ⊕ ROT R13 (x) ⊕ ROT R22 (x)

**Función**

Es una función que realizará operaciones lógicas y operaciones binarias rotativas tomando como parámetros de entrada la variable Ei. Su fórmula es la siguiente:

(x) = ROT R6 (x) ⊕ ROT R11 (x) ⊕ ROT R25 (x)

Este proceso se llevaría a cabo durante 64 rondas. Una vez hubiéramos terminados estas rondas sumaríamos los valores que tengan cada una de las variables A,B,C,D,E,F,G,H con el valor que hayan tenido en la primera ronda y junto con el valor en la última ronda, esta suma será en módulo 32. Por último, obtendríamos nuestro hash concatenando el valor de las variables A,B,C,D,E,F,G,H.

Por ejemplo, en el caso de calcular el hash para las palabras Gonzalo Nazareno las sumas serían las siguientes:

A0 => 01101010000010011110011001100111

+

A63 => 01101011011111011101100001010100

11010101100001111011111010111011 => Hexadecimal => D587BEBB

B0 => 10111011011001111010111010000101

+

B63 => 11101010100000111101000100001011

10100101111010110111111110010000 => Hexadecimal => A5EB7F90

C0 => 00111100011011101111001101110010

+

C63 => 11010010010000101000011010010011

00001110101100010111101000000101 => Hexadecimal => 0EB17A05

D0 => 10100101010011111111010100111010

+

D63 => 10101010011011010100110111110011

01001111101111010100001100101101 => Hexadecimal => 4FBD432D

E0 => 01010001000011100101001001111111

+

E63 => 11101101100011101000110011001000

00111110100111001101111101000111 => Hexadecimal => 3E9CDF47

F0 => 10011011000001010110100010001100

+

F63 => 11000000001010110000001011010011

01011011001100000110101101011111 => Hexadecimal => 5B306B5F

G0 => 00011111100000111101100110101011

+

G63 => 00100111000111110000011000010111

01000110101000101101111111000010 => Hexadecimal => 46A2DFC2

H0 => 01011011111000001100110100011001

+

H63 => 01100110011011111001101011000101

11000010010100000110011111011110 => Hexadecimal => C25067DE

Una vez tenemos el resultado de las ocho sumas tan sólo nos quedara concatenar los valores para obtener el hash de Gonzalo Nazareno.

Gonzalo Nazareno => A+B+C+D+E+F+G+H => D587BEBBA5EB7F900EB17A054FBD432

D3E9CDF475B306B5F46A2DFC2C25067DE

En caso de que el lector tenga interés por esta tecnología he diseñado una aplicación web la cual calcula automáticamente el hash de la palabra que se introduzca. Para poder acceder a ella puede dirigirse al siguiente enlace: <https://jesusrubio3.github.io/hash/>

**Libro mayor**

El libro mayor, también conocido como ledger, es aquel que registra todas las operaciones o transacciones, sin excepción, que se han realizado en la blockchain desde su bloque génesis hasta la actualidad. Es por ello, que el ledger es unos de los elementos fundamentales para la transparencia, seguridad y privacidad en la que se respalda las cadenas de bloques.

Este libro mayor ha sido utilizado por la humanidad desde la aparición de la escritura. Se tiene constancia de que en el Antiguo Oriente Próximo ya eran usados para registrar datos como la producción, el comercio y las deudas. Estos libros han sufrido bastantes evoluciones, una de las más importantes surge en el siglo XIX mediante el cual estos libros pasan a estar **centralizados** debido al avance de las grandes burocracias. Por lo cual ya resolvimos uno de los conflictos, ya no iban a existir problemas de inconsistencia de datos debido a que solo iba haber un “único libro mayor” basado en la confianza depositadas por los ciudadanos en las instituciones centralizadas.

Un siglo más tarde, debido al avance de la tecnología informática, los ledger sufren otra evolución. Aquellos libros escritos en papel iban a pasar a formato digital evitando así perder los datos por causas como incendios, inundaciones o terremotos. Pero hoy día seguimos teniendo el problema que estos libros mayores, convertidos en bases de datos o grandes ficheros, siguen estando centralizados. Digitalizar la información nos permite un análisis, cálculo y velocidad de cómputo incomparable con las versiones en papel, pero estas bases de datos son tan fiables como la organización y las personas por las que son mantenidos.

Es aquí otro de los problemas que se pueden resolver gracias al blockchain. La tecnología de las cadenas de bloques está basada en la filosofía de los libros mayores, pero resolviendo el problema de la dependencia de una autoridad central en la que tengamos que depositar nuestra confianza y sea esta la única responsable de mantener y validar el libro mayor.

Aunque blockchain y ledger son en cierto modo parecidos no podemos caer en el error de creer que son la misma tecnología. El libro mayor es una base de datos en lo que almacenamos toda la información de la red. Esta base de datos no va a estar centralizada en un único servidor, sino que cada nodo de nuestra red va a guardar una copia de la información que almacena dicha base de datos consiguiendo así que sea muy difícil que nuestra red sea atacada con éxito, pues el atacante debería tener la posición de más del 50% de los nodos que forman nuestra red.

Por la otra parte tenemos blockchain que es un tipo de libro mayor segurizado mediante la criptografía y el enlace de los bloques al poseer información del bloque inminentemente anterior. Por lo tanto, conseguimos que si alteramos un solo bits de unos de nuestros bloques de un nodo los demás nodos comprobarán que dicho bloque de dicho nodo tiene una información errónea pues no concuerda con la información que posee la mayoría y automáticamente la base de datos de ese nodo será actualizada.

Los ledgers que podemos encontrarnos pueden ser de dos tipos:

* Ledgers centralizados, los cuales son controlados y validados por una sola entidad. Un ejemplo podría ser el diseño de los bancos o los estados gubernamentales.
* Ledgers distribuidos, pueden ser duplicados y almacenados por cualquier nodo. Su seguridad está asegurada debido a que para introducir un dato debe existir un consenso entre la mayoría de los nodos. Evitamos que una sola entidad posea el control absoluto pues su fuerza posee en la red, la seguridad de la red es directamente proporcional al número de dispositivos que posee la base de datos. Una blockchain con muchos bloques es una cadena más segura.

Por lo tanto, en el caso de que un ciberdelincuente quisiera alterar la información de la base de datos de nuestra cadena, es decir, los datos que contiene uno de nuestros bloques, debería asegurarse de modificar el hash del bloque que va a corromper pues al permutar el contenido cambiará el hash.

Supongamos que lo consigue. El ciberdelincuente ha logrado crackear nuestro sistema y modificar la información de un bloque y su hash, pero le surge un nuevo problema, al modificar el hash de dicho bloque la cadena se corrompe ya que el bloque posterior no le cuadra el valor del hash previo, el bloque que ha sido alterado. Vemos aquí la potencia de la seguridad que guarda la tecnología de bloques, no solo tendría que modificar el hash de ese bloque sino el valor de todos los hashes posteriores a ese bloque.

Aunque el lector se puede dar cuenta que en el ejemplo anterior no estamos exponiendo la ventaja fundamental de la que hemos hablado en este apartado. Una cadena de bloques puede ser más difícil de alterar que una base de datos convencional o un fichero Excel, pero seguimos teniendo el mismo problema si no es más respaldada por otros nodos. Es por ello que la gran mayoría de blockchain usan el sistema peer-to-peer.

<https://academy.bit2me.com/que-es-ledger-distribuido-libro-mayor/>

<https://www.innovaciondigital360.com/blockchain/que-es-y-como-funciona-la-tecnologia-de-libros-distribuidos-blockchain-dlt/>

<https://medium.com/cryptoeconomics-australia/the-blockchain-economy-a-beginners-guide-to-institutional-cryptoeconomics-64bf2f2beec4>