Estudio y Investigación de Ethereum

Elvi Mihai Sabau Sabau[1][emss5]

[1] Universidad de Alicante, Alicante, España.
frenzoid@pm.me

Abstract. En este informe hablaremos sobre Ethereum, que es, cómo funciona, qué costes tiene, que tecnologías se usa para desarrollar dapps, y también compararemos algunas block chains basadas en Ethereum.

Keywords: Ethereum, Solidity, Dapps, DLT, Blockchain.

1 Introducción.

1.1 Que es Ethereum.

Ethereum es una blockchain open source basada en DLT, con su propia criptomoneda, y con la capacidad de almacenar datos y ejecutar lógica en sus nodos.

Esta última característica es la que la diferencia de la mayoría de black chains de hoy en día, además de la posibilidad de poder clonar el proyecto, y desplegar una versión propia de Ethereum.

2 EVM

La EVM (Ethereum Virtual Machine)*[1]* es el motor de Ethereum, se encarga de almacenar todos los datos y de ejecutar las operaciones de los contratos inteligentes.

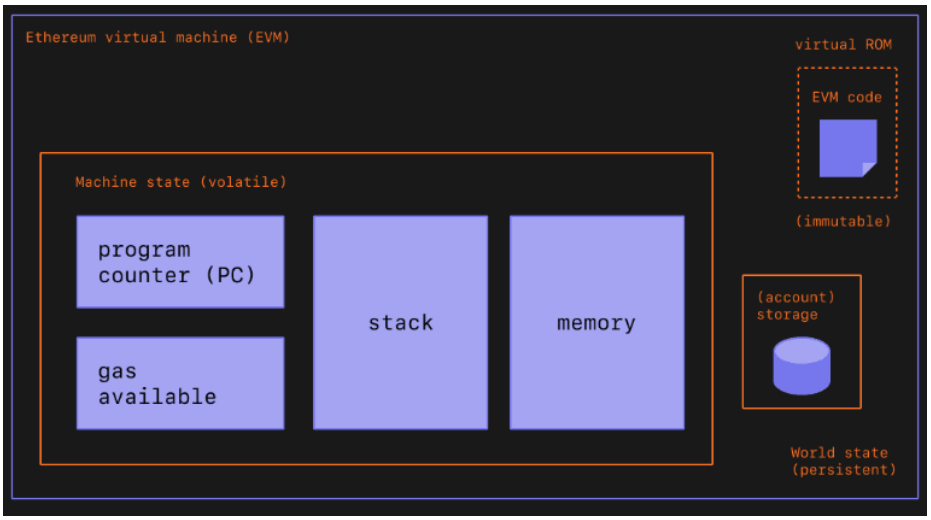

Fig. 1 Esquema de la EVM.

El EVM se comporta como lo haría una función matemática, dada una entrada, produce una salida determinista. Por lo tanto, es muy útil describir de manera más formal a Ethereum como si fuese una función de transición de estados.

2.1 Contexto.

En el contexto de Ethereum, el “state” es una enorme estructura de datos Merkle Patricia Trie modificada*[2]*, que mantiene todas las cuentas vinculadas por hashes y se puede reducir a un solo hash raíz almacenado en la blockchain.

2.2 Instrucciones.

La EVM se encarga de ejecutar instrucciones definidas *(OPCODES)[3][4]*, que se ejecutan en la blockchain.

Para evitar que hayan bucles infinitos u otras instrucciones capaces de agotar los recursos de los nodos de la blockchainada, y para incentivar a los mineros a minar las transacciones que realiza una instrucción, cada instrucción tiene un coste en ETH definido en una unidad llamada GAS.

2.3 Gas.

El gas es una unidad de medida del ETH, que se usa para medir cuánto ETH cuesta ejecutar una operación.*[5]*

$$gasUsed * gasPrice = gasFee$$

Fig. 2 Fórmula del cálculo del gas.

Dado que cada transacción de Ethereum requiere recursos computacionales para ejecutarse, cada transacción requiere una tarifa.

Gas se refiere a la tarifa requerida para realizar una operación en Ethereum con éxito.

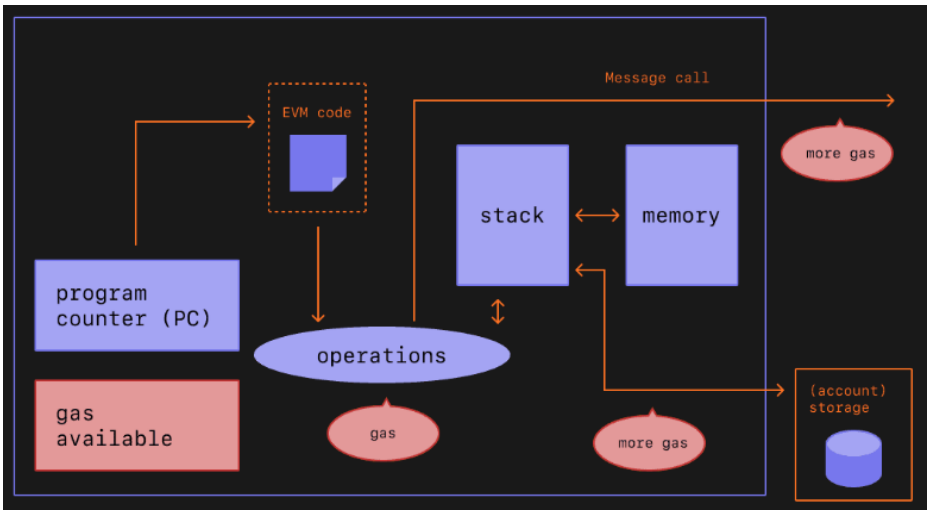

Fig. 3 Esquema de interconexión de los módulos internos de la EVM.

Debido a la variación de la carga de trabajo de los nodos de la blockchain, el precio del gas suele oscilar durante el día.

Hay páginas como Etherescan*[17]* que ayudan a estimar el coste del gas del día.

El cálculo de la tarifa de las transacciones en gas funciona de la siguiente manera:

```
unidades de gas (límite) * (tarifa base + propina)
```

Cuando un usuario ejecuta una instrucción, éste puede definir el límite de gas a usar, de esta manera, si hay gas sobrante (ETH que no se quema al ejecutar las instrucciones) el usuario puede elegir solicitar la devolución del cambio, o usarlo para dar prioridad a la ejecución de dicha transacción, siendo este gas sobrante un incentivo extra para los mineros.

Esto se puede representar en la siguiente fórmula:

```
reembolso = tarifa máxima - (tarifa base + tarifa
prioritaria)
```

Por otro lado, cuando no hay gas suficiente, este gas se paga a los mineros por finalizar transacciones. Incluso si falla, los mineros deben validar y ejecutar la transacción.

3 Nodos y tipos de nodos.

Los nodos son donde se ejecutan las operaciones y transacciones de la blockchain.
En Ethereum, tenemos 3 tipos de nodos.*[6]*

3.1 Completo.

- Almacena datos completos de blockchain.
- Participa en la validación de bloques, verifica todos los bloques y estados.
- Todos los estados se pueden derivar de un nodo completo.
- Sirve a la red y proporciona datos a pedido.

3.2 Ligero.

- Almacena la cadena de encabezado y solicita todo lo demás.
- Puede verificar la validez de los datos contra las raíces estatales en los encabezados de bloque.
- Útil para dispositivos de baja capacidad, como dispositivos integrados o teléfonos móviles, que no pueden permitirse almacenar gigabytes de datos de blockchain.

3.3 Archivo.

- Almacena todo lo que se mantiene en el nodo completo y crea un archivo de estados históricos.
- Útil para acceder a transacciones concretas de hace tiempo.

4 Tipos de consenso.

Los consensos (también conocidos como protocolos de consenso o algoritmos de consenso) permiten que los sistemas distribuidos trabajen juntos y permanezcan seguros.

4.1 Proof of Work.

El PoW es una forma de prueba criptográfica en la que una parte demuestra a los demás (los verificadores) que se ha gastado una cierta cantidad de esfuerzo computacional en una operación específica.

4.2 Proof of Stake.

Las PoS son un tipo de consenso concretamente para block chains que funcionan seleccionando validadores en proporción a su cantidad de tenencias en la criptomoneda asociada. Esto se hace para evitar el costo computacional de los consensos PoW.

4.3 Otros.

Otras block chains que también usan el EVM pero que no pertenecen a la mainnet pueden implementar otros tipos de consenso.

5 Redes.

En Ethereum, hay diferentes tipos de redes blockchain. La mainnet, y las testnets.

5.1 Mainnet.

La red Mainnet es la principal, y es donde se despliegan los contratos en producción.

5.2 Testnets.

Las testnets son redes donde la gente puede desplegar sus contratos para probarlos, la criptomoneda de estas testnets no valen nada, y sirven únicamente para el propósito de probar el correcto funcionamiento del contrato a desplegar. Un ejemplo de esto es Kovan. *[29]*.

6 Tipos de Cuentas.

En ethereum hay 2 tipos de cuentas.*[7]*

6.1 Propiedad Externa.

Son aquellas que se usan para carteras y representan una persona en la blockchain, es gratuita de crear, y tienen claves publicas y privadas para firmar las transacciones.

6.2 Contratos.

Son aquellas que se usan para los contratos, ya que los contratos también tienen una dirección y una cartera para guardar, y cuesta cierta cantidad de ETH crearla, debido a que al desplegar el contrato, se debe alojar cierta cantidad de memoria del state de la EVM donde alojar el bytecode del contrato, no tienen claves privads ni publicas porque requieren de una interaccion externa para realiar una transacción..

7 Contratos.

Los contratos son aplicaciones en bytecode basadas en instrucciones de la EVM que se ejecutan en la blockchain. Los Contratos son muy similares a las Clases en la programación orientada a objetos.

7.1 Lenguajes de programación.

El código de los contratos inteligentes generalmente se escribe en un lenguaje de programación de alto nivel como Solidity. *[9]*

También se puede desarrollar en otros lenguajes, como Python usando Vyper, o C + +. Pero es recomendable usar Solidity por la sencillez y las capacidades de manejo de las instrucciones.

Este código se compila (usando el compilador solc) en algo llamado bytecode EVM que se implementa en la cadena de bloques de Ethereum.

Esto es muy similar a un lenguaje de programación como Java, donde el código se convierte en código JVM Byte.

Un ejemplo simple que de un contrato inteligente en Solidity que puede obtener, incrementar y decrementar un contador.*[8]*

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;

contract Counter {
    uint public count;

    // Function to get the current count
    function get() public view returns (uint) {
        return count;
    }

    // Function to increment count by 1
    function inc() public {
        count += 1;
    }

    // Function to decrement count by 1
    function dec() public {
        count -= 1;
    }
}
```

Fig. 4 Ejemplo en Solidity de un programa sencillo.

También, añado una chuleta de solidity *[10]* desde ejemplos sencillos, hasta conceptos complejos.

7.2 OpenZeppelin.

Openzeppelin*[11]* es una librería de contratos predefinidos, que sirven como plantillas para tokens o NTFs, o funcionalidades extra o de verificación como SafeMath*[12]* que sirven como contenedores sobre operaciones aritméticas de Solidity con comprobaciones de desbordamiento de bits.

Para implementarlos en el contrato que se está desarrollando solo se tiene que heredar del contrato de open zeppelin.

8 Entornos de desarrollo.

Cuando se desarrollan contratos inteligentes, es indispensable tener un buen entorno de desarrollo.

La manera clásica de desarrollar contratos es la de usar el compilador “solc” y desplegar la aplicación a la testnet desde algún endpoint, y después de probarla, desplegarla nuevamente a la mainnet, pero esta manera de desarrollo es antigua e ineficiente.

A continuación hablaremos sobre varias herramientas que nos sirven para desarrollar de forma óptima contratos inteligentes en Solidity.

8.1 Ganache y Truffle.

Ganache*[13]* y Truffle*[14]* son dos frameworks que se usan para simular una blockchain en nuestro ordenador local, donde desplegar y testear nuestro contratos, además nos sirven para conectarnos a los endpoints de la blockchain de Ethereum y desplegar el bytecode (contrato compilado) a la red.

8.2 Hardhat.

Hardhat es lo mismo que Ganache y Truffle, pero todo en uno, además de permitir el testeo de dicho contrato en la red local usando Mocha, y un despliegue más limpio que Truffle.

Hardhat está mejor documentada que Truffle y Ganache, y se usa más en entornos profesionales, pero es más complejo de usar, y debido a que la herramienta se actualiza cada día, a veces la documentación puede ser liante.

Truffle y Ganache son más sencillos de usar, pero son más limitados, ya que no son tan configurables.

9 Endpoints.

Concretamente, un endpoint es una URL que permite la interacción con la blockchain mediante métodos HTTP. Es como una API de la blockchain.

Los endpoints son accesos a los Nodos de Ethereum, un endpoint suele ser un servicio por el cual se puede acceder a la red de Ethereum mediante un nodo en concreto.

Ethereum tiene varios Nodos, y hay varias empresas / organizaciones que comercializan el acceso a la blockchain, alquilando su endpoint.

9.1 Infura.*[15]*

Infura es una de esas empresas que comercializan el acceso a su nodo mediante una API key, es muy limitada, pero es la más famosa.

9.2 Moralis.*[16]*

Moralis es otra entidad que permite el acceso a su endpoint, pero de forma gratuita.

10 Carteras.

Las carteras son solo aplicaciones que nos sirven para interactuar con las cuentas externas.

10.1 Metamask.*[18]*

Metamask es una de las más conocidas, se integra con el navegador, y se enlaza directamente con la Dapp a usar mediante el navegador web.

11 Dapps.

Las DAPPs o (Decentralized APP) son aplicaciones conectadas a un smart contract en la blockchain como por ejemplo Open Sea*[19]* y Crypto Kitties*[20]*.

Se usa un frontend cualquiera con una librería capaz de conectarse a un endpoint y a la cartera integrada (metamask) del usuario.

Hay una alta tendencia últimamente en el mercado, la mayoría de desarrolladores de DAPPs usan ReactJS para el frontend, usando Web3JS o EthersJS para conectarse al endpoint para poder interactuar con el contrato, y para conectarse a la cartera integrada para poder interactuar con el usuario.

11.1 Web3JS / EthersJS.

Web3JS*[21]* y EthersJS*[22]* son librerías en JavaScript que sirven para conectar un frontend en JavaScript con un endpoint en concreto y para poder interactuar con las carteras integradas de los usuarios.

11.2 Scaffold-Eth. *[23]*

Scaffold-Eth es un proyecto opensource que permite al desarrollador de DAPPs, desplegar un proyecto usando Hardhat y React. La magia es que los componentes de React se actualizan en tiempo real dependiendo de los métodos que posee el contrato.

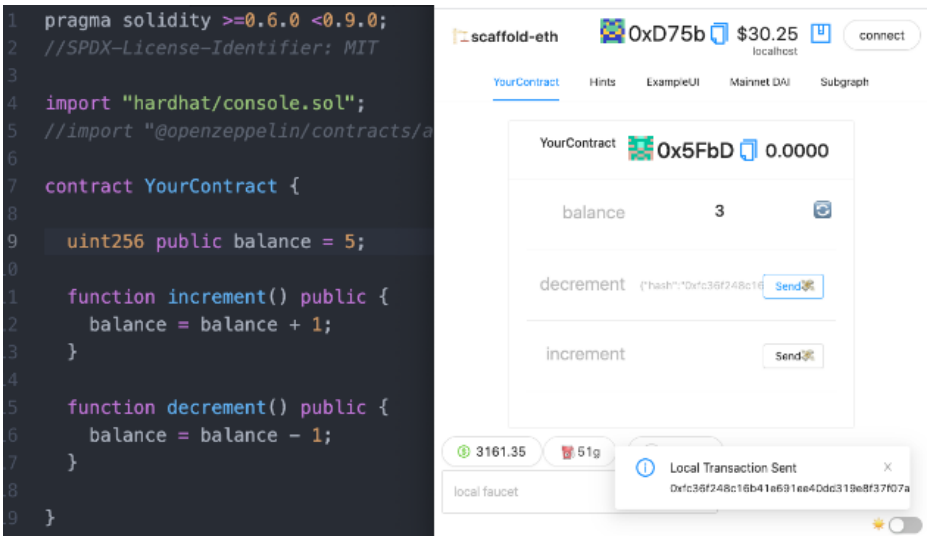

Fig. 5 Ejemplo de Scaffold-Eth, a la izq está el contrato, y a la derecha está la app en React con un componente autogenerado por cada campo / método del contrato.

De esta manera, se hace increíblemente fácil probar y desarrollar una interfaz para nuestra DAPP.

12 Blockchains Derivadas.

Como hemos dicho originalmente, ethereum es opensource, y permite que cualquier persona con los recursos hardware y software necesarios, pueda desplegar su propia implementación de Ethereum.

Debido a esto, hay más de 100 block chains diferentes, en chainlist.org*[24]* podemos ver una lista de todas estas las cuales son de uso público, todas estas son derivadas de ethereum (sus nodos ejecutan una EVM con una configuración propia).

12.1 Crea tu propia blockchain.

Para crear nuestra propia blockchain basada en Ethereum*[26][29][30]*, podemos usar GETH *[25]* (Go Ethereum) una implementación de la EVM en Go. O forkear el proyecto original de Ethereum desde github y modificarlo, esta está desarrollada en C + +. *[27]*

Lo único que principalmente debemos hacer es cambiar la ChainID de nuestro nodo, de tal manera que no esté enlazada con ninguna otra blockchain, siendo la “1” la principal.*[26]*

Lo bueno de crear una blockchain propia es que es totalmente gratuito mientras tengas los recursos hardware necesarios, el coste del despliegue y mantenimiento es gratuito, y las tarifas son configurables al igual que el precio y la cantidad de la criptomoneda.

12.2 Harmony. *[31]*

Harmony es otra blockchain basada en EVM, la diferencia es que es extremadamente barata en comparación con el resto de block chains del estilo, además de permitir el envío de datos entre block chains, y de ser bastante eficiente al tener un alto tps.

12.3 xDAI. *[32]*

xDAI es otra blockchain basada en EVM, la diferencia es que usa otro tipo de consenso PoA (Proof of Authority) y que el precio de su moneda es casi estable, fluctúa muy poco.

12.4 Telos. *[33]*

Telos es otra blockchain basada en EVM, esta es una de las más maduras y de las que más bloques ha producido comparado con el resto de block chains, además sus tarifas son muy bajas ya que su moneda es bastante barata (0.887$ = 1 Telos), además usa un consenso modificado del PoS, DPoS (Delegated Proof of Stake).

13 Tablas de costes.

A continuación mostraremos una pequeña tabla comparativa de los costes del gas medio, los costes en gas de las tarifas de transacción, y el coste de la moneda en USD.

Blockchain	1 Gas	Gas usado en transacciones	Coste de la moneda
Ethereum	0.0014 ETH - 0.0100 ETH	21000	~$2,557.77
Harmony	~0.000021 ONE	21000	~$0.1239
xDai	~0.00005 USDI	40000	~$1.003
Telos	.~0.01 TLOS (byte)*[34]*	21000	~$0.860

14 Competencia.

Actualmente, una de las mejores competencias de Ethereum es Solana*[35]*, una blockchain en la cual los contratos se desarrollan en C, o Rust.

Otro férreo competidor es Cardano*[36]*, y curiosamente, el lenguaje de programación de los contratos es Haskell.

15 Polygon

Aparte de la existencia de Blockchain que buscan ser competidoras de Ethereum, como puede ser Solana, también existen otras que buscan mejorar algunos aspectos de

Ethereum. Uno de estos proyectos es Polygon, que es un proyecto que constituye un conjunto de soluciones para proporcionar una mayor escalabilidad a Ethereum.

Una de las soluciones más conocidas de Polygon es la implementación de las sidechain mediante Polygon PoS.

15.2 Sidechains

Las sidechains son 'clones' de una blockchain padre, que en este caso será Ethereum. Entre estas blockchain se crea un sistema que permite comunicarse entre sí. Este sistema se conoce como 'puente' (bridge), que permite mover activos entre estas.

El concepto de las sidechain es que, haciendo uso de estas, se pueden minimizar la cantidad de transacciones producidas en la blockchain padre. Cuando se desea mover activos a la sidechain, se 'bloquean' los activos escogidos dentro de la blockchain padre mediante un smart contract y son creados dentro de la sidechain, siendo asignados a los activos de la blockchain padre.

Una vez creados los activos, se pueden mover los activos creados como se desee, hasta que se deseen extraer los activos. Para extraerlos, se 'eliminarán' los activos de la sidechain y se desbloquearán de la blockchain padre.

De esta forma, al final de todo solo serán realizadas 2 transacciones en la blockchain principal: El bloqueo y el desbloqueo de los activos, lo que disminuye muy significativamente la cantidad de transacciones finales en la principal.

15.3 Puente (Bridge)

Es el sistema encargado de controlar la capacidad de mover activos entre las 2 blockchains. En Polygon se pueden encontrar principalmente 2 tipos de bridges. Estos son:

- PoS Bridge: El bridge más recomendado debido a la flexibilidad y velocidad de retiros de activos que posee.
- Plasma bridge: Bridge que ofrece una mayor seguridad a cambio de flexibilidad.

15.4 Validadores y delegadores

En Polygon, para poder participar en el consenso PoS, puedes participar de 2 formas:

- Siendo un validador, que posee un nodo validador y se encarga de realizar las validaciones de las transacciones de la red, y obtener las recompensas de estas. Este nodo validador 'apostará' una cantidad de activos para asegurar su validez, ya que en el caso de que su predicción no sea correcta, se le quitará esa cantidad de activos, repartiéndola a los nodos que hayan acertado la validación.

- Siendo un delegador, que no posee un nodo, solo activos. Este delegador se encarga de 'apostar' a un nodo validador de su elección. Si el nodo validador acierta, la recompensa se dividirá entre este y los delegadores que posea, mientras que si falla, la pérdida se repartirá también entre el nodo y los delegadores.

15 Referencias.

1. EVM, Ethereum.org, https://ethereum.org/en/developers/docs/evm/ , accedido el 11/03/2022.
2. patricia-trie, eth.wiki, https://eth.wiki/fundamentals/patricia-tree, accedido el 11/03/2022
3. Lista de instrucciones Excel, docs.google.com, https://docs.google.com/spreadsheets/d/1n6mRqkBz3iWcOlRem_mO09GtSKEKrAsfO7Frgx18pNU/edit#gid=0, accedido el 11/03/2022
4. Opcodes, Ethereum.org, https://ethereum.org/en/developers/docs/evm/opcodes, accedido el 11/03/2022
5. Gas, Ethereum.org, https://ethereum.org/en/developers/docs/gas/, accedido el 11/03/2022
6. Nodos, Ethereum.org, https://ethereum.org/en/developers/docs/nodes-and-clients/, accedido el 11/03/2022
7. Acounts, Ethereum.org, https://ethereum.org/en/developers/docs/accounts/, accedido el 11/03/2022
8. Solidity By Example, https://solidity-by-example.org/, accedido el 11/03/2022
9. Solidity docs, https://docs.soliditylang.org/en/v0.8.12/, accedido el 11/03/2022
10. Frenzoid's Github, Github, https://github.com/Frenzoid/labs/blob/master/SOLIDITY/0_solidity_megacheatsheet.sol, accedido el 11/03/2022
11. OpenZeppelin, OpenZeppelin, https://docs.openzeppelin.com/ accedido el 11/03/2022
12. SafeMath, OpenZeppelinl, https://docs.openzeppelin.com/contracts/2.x/api/math, accedido el 11/03/2022
13. Ganache, https://trufflesuite.com/ganache/index.html, accedido 11/03/2022
14. Truffle, https://trufflesuite.com/truffle/, accedido el 11/03/2022
15. Infura, https://infura.io/, accedido el 11/03/2022
16. Moralis, https://moralis.io/, accedido el 11/03/2022
17. Etherscan, https://etherscan.io/gastracker, accedido el 11/03/2022
18. Metamask, https://metamask.io/, accedido el 11/03/2022
19. OpenSea, https://opensea.io/, accedido el 11/03/2022
20. CryptoKitties, https://www.cryptokitties.co/, accedido el 11/03/2022
21. Web3JS, https://web3js.readthedocs.io/en/v1.7.1/, accedido el 11/03/2022
22. EthersJS, https://docs.ethers.io/v5/, accedido el 11/03/2022
23. Scaffold-Eth, https://github.com/scaffold-eth/scaffold-eth, accedido el 11/03/2022
24. ChainLink, https://chainlist.org/, accedido el 11/03/2022
25. GoEthereum, https://geth.ethereum.org/docs/install-and-build/installing-geth, accedido el 11/03/2022
26. GoEthereum, https://geth.ethereum.org/docs/interface/private-network, accedido el 11/02/2022
27. Merehead, https://merehead.com/blog/how-to-create-private-ethereum-blockchain/, accedido el 11/02/2022
28. Kovan, https://kovan-testnet.github.io/website/, accedido el 11/03/2022
29. europa.eu, https://joinup.ec.europa.eu/collection/blockchain-egov-services/document/user-guide-how-setup-private-ethereum-poa-blockchain-network, accedido el 11/03/2033
30. europa.eu, https://joinup.ec.europa.eu/collection/blockchain-egov-services, accedido el 11/03/2022
31. Harmony.one, documentacion, https://docs.harmony.one/home/, accedido el 11/03/2022
32. xDay, documentacion, https://www.xdaichain.com/for-developers/developer-resources, accedido el 11/03/2022
33. Telos, documentacion, https://docs.telos.net/evm/about-ethereum-virtual-machine/comparing-telos-to-other-evm-chains, accedido el 11/03/2022
34. Telos, gas price per byte estimate, https://docs.telos.net/evm/about-ethereum-virtual-machine/gas-fees accedido el 11/03/2022
35. Solana, https://solana.com/es, accedido el 11/03/2022
36. Cardano, https://cardano.org/, accedido el 11/03/2022