Sergi Roviralta Suárez

ANÀLISI I ESTUDI DE LA TECNOLOGIA BLOCKCHAIN AMB EL DESEVOLUPAMENT D'UNA DAPP

TREBALL DE FI DE GRAU

dirigit per Marc Sánchez Artigas

Grau d'enginyeria informàtica

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2023

Resum

La blockchain o cadena de blocs és una tecnologia que actualment està agafant molta força en l'àmbit de desenvolupament web i mòbil. En aquest treball es farà principalment un estudi d'aquesta tecnologia juntament amb una anàlisi de l'estat d'art.

Es començarà marcant els objectius i la motivació per a fer aquest treball continuant amb l'explicació de l'evolució del blockchain i les seves característiques més importants que fan que tingui tant de projecte i estigui en constant evolució.

La següent secció es dedicarà al desenvolupament i a entendre una de les plataformes més importants actualment: Ethereum. Es centrarà sobretot com és el seu funcionament, estructura i els smart contracts, permetent així automatitzar processos garantint una transparència i seguretat que avui en dia és tan buscada.

Seguidament, i la part més rellevant, és l'anàlisi que es duu a terme d'aplicacions reals per a l'estudi de diferents plataformes blockchain i a més, l'estudi de casos d'ús reals per a veure el seu rendiment i eficiència en aquestes aplicacions.

Un cop realitzada la teoria d'aquestes tecnologies, s'acabarà amb la implementació d'una DAPP (aplicació descentralitzada) per a posar en pràctica els coneixements apresos durant la recerca i investigació del treball. D'aquesta manera assolir coneixements del llenguatge de programació solidity per a saber interactuar amb els contractes intel·ligents.

Resumen

La blockchain o cadena de bloques es una tecnología que actualmente está cogiendo mucha fuerza en el ámbito de desarrollo web y móvil. En este trabajo se realizará principalmente un estudio de esta tecnología junto con un análisis del estado de arte.

Se empezará marcando los objetivos y la motivación para realizar este trabajo continuando con la explicación de la evolución del blockchain y sus características más importantes que hacen que tenga tanto proyecto y esté en constante evolución.

La siguiente sección se dedicará al desarrollo y a entender una de las plataformas más importantes actualmente: Ethereum. Se centrará sobre todo cómo es su funcionamiento, estructura y los smart contracts, permitiendo así automatizar procesos garantizando una transparencia y seguridad que hoy en día es tan buscada.

Seguidamente, y la parte más relevante, es el análisis que se lleva a cabo de aplicaciones reales para el estudio de diferentes plataformas blockchain y además, el estudio de casos de uso reales para ver su rendimiento y eficiencia en estas aplicaciones.

Una vez realizada la teoría de estas tecnologías, se acabará con la implementación de una DAPP (aplicación descentralizada) para poner en práctica los conocimientos aprendidos durante la investigación del trabajo. De esta forma alcanzar conocimientos del lenguaje de programación solidity para saber interactuar con los contratos inteligentes.

Abstract

Blockchain is a technology that is currently gaining momentum in the field of web and mobile development. In this project we will mainly carry out a study of this technology together with an analysis of the state of the art.

It will start by setting the objectives and motivation for this project, continuing with the explanation of the evolution of the blockchain and its most important characteristics that make it have so much project and is constantly evolving.

The next section will be dedicated to the development and understanding of one of the most important platforms today: Ethereum. It will focus on how it works, its structure and smart contracts, allowing to automate processes and guaranteeing the transparency and security that is so sought after nowadays.

Next, and the most relevant part, is the analysis of real applications for the study of different blockchain platforms and the study of real use cases to see their performance and efficiency in these applications.

Once the theory of these technologies is done, it will end with the implementation of a DAPP (decentralised application) to put into practice the knowledge learned during the research work. In this way, knowledge of the solidity programming language will be acquired in order to know how to interact with smart contracts.

Índex

Índex de figures

Índex de taules

1. Introducció

1.2 Motivació

La motivació per fer aquest treball és la creixent importància que està adquirint la tecnologia blockchain en la nostra societat actual. La blockchain té un gran potencial en molts àmbits, però el d'avui serà el desenvolupament web i mòbil.

Així doncs, és important comprendre el funcionament de la tecnologia blockchain, els seus avantatges i limitacions, així com les seves diferents aplicacions pràctiques. A més, amb el desenvolupament d'aplicacions descentralitzades, és essencial conèixer les eines i els entorns de desenvolupament necessaris per a la seva creació.

1.2 Objectius

L'objectiu d'aquest treball és proporcionar un estudi complet de la tecnologia blockchain i les seves diferents aplicacions pràctiques, entendre com funciona i quins són els avantatges i desavantatges que proporciona a la nostra societat. Així com trobar quina és la millor plataforma realitzant una anàlisi del rendiment i eficiència d'aplicacions pràctiques.

Desenvolupar una aplicació descentralitzada posant en pràctica els conceptes teòrics apresos durant tot el projecte amb l'objectiu de fomentar el seu ús en un futur i aprendre les eines i llenguatges de programació utilitzades durant aquest procés.

Aquest objectiu serà implementar un sistema de votació descentralitzat, permetent el vot a usuaris fent servir el seu compte d'una cartera de criptomonedes com pot ser Metamask.

2. Blockchain

2.1 Evolució històrica de la blockchain

La tecnologia que sustenta la cadena de blocs no va sorgir d'un dia per l'altre, sinó que és la conseqüència a més de 40 anys d'investigacions.

Una de les peces fonamentals que hi ha darrere la tecnologia blockchain és la criptografia. Aquesta s'encarrega de codificar les dades, de manera que només es poden descodificar si es té possessió de la clau corresponent. En el punt 2.5 es detallarà més el tema, però per ara es parlarà de dos conceptes:

- **Xifrat asimètric o de clau pública:** tècnica de xifratge que utilitza un algorisme de clau pública o provada per a la comunicació de dades segura. Consisteix en el xifratge d'un missatge fent servir la clau pública del destinatari. Perquè el destinatari pugui desxifrar aquest missatge només pot utilitzar la seva clau privada. És a dir que és un sistema en el qual s'usen les dues claus.

- **Funció hash:** és un tipus de mecanisme de seguretat que produeix un valor de hash, una digestió de missatges o un valor de comprovació per a un objecte de dades específic. S'implementen per a avaluar la integritat de les dades, el control d'autenticació i altres mecanismes de seguretat.

El desenvolupament de la tecnologia blockchain no hauria estat possible sense aquestes dues tècniques de xifratge desenvolupades en la dècada dels 1960 i 1970.

Per què es va crear blockchain? Stuart Haber i W. Scott Stornetta van treballar en una cadena de blocs protegida criptogràficament en la que ningú podia manipular les marques de temps dels documents, és a dir el que no hi havia era un sistema que certifiqués la data en la qual s'havia generat el document.

L'any 1991 van publicar finalment el resultat de la seva investigació on la solució era usant una cadena de blocs. Els documents s'encadenaven i emmagatzemaven un darrere de l'altre formant una seqüència de temps immutable.

Les primeres propostes a l'ús de la blockchain van ser el treball de l'enginyer informàtic Wi Dai on va publicar un treball sobre com crear una criptomoneda tractant la descentralització del sistema i de la protecció de l'anonimat.

Per altra banda, l'any 1998 el científic informàtic Nick Szabo va publicar un document en el qual proposava la creació d'un sistema monetari descentralitzat al qual va dir Bit Gold. La seva particularitat es que el valor de les seves monedes es basava en el cost dels recursos utilitzats (cost computacional). Per aquesta mateixa raó no es va acabar implementant, ja era una incongruència.

Però és l'any 2008 que la història comença a guanyar rellevància gràcies al treball d'una persona o grup amb el nom de Satoshi Nakamoto. Va publicar el primer informe sobre la tecnologia, proporcionant detalls sobre un sistema electrònic de pagament descentralitzat entre parells: *Bitcoin, a peer to peer electronic cash System.*

El document es va publicar amb el pseudònim de Saoshi Nakamoto i encara no se sap qui hi ha darrere d'aquest nom. Es tractava d'una moneda virtual que no estava controlada per ningú, és a dir, que era una moneda descentralitzada, el que es coneix avui en dia com a criptomoneda.

El sistema de Bitcoin funciona que cada vegada que es traspassa la propietat de les monedes, l'operació queda registrada en el registre públic de la blockchain. Confirmant d'aquesta manera un sistema immutable i evitar registre de transaccions fraudulentes. Aquestes transaccions són validades per a multitud de participants (ordinador miners) que estan fent càlculs perquè al final sol un s'emporti la recompensa per a validar el bloc, aquest sistema es coneix com a prova de treball (POW).

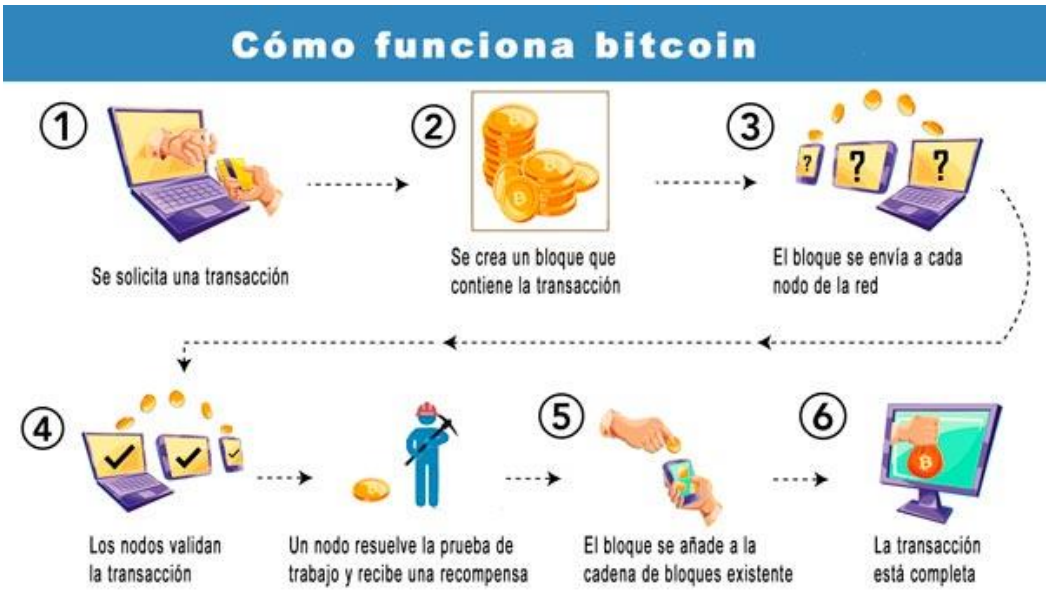


Figura 1: ***:*** *Imatge mostrant el procés d'una transacció Bitcoin. [1]*

Semblava que la tecnologia blockchain era ideal, però va continuar evolucionant quan va sorgir una altra plataforma l'any 2015 amb el nom d'*Ethereum.* Aquesta albergava el registre de contractes intel·ligents, programes que s'implementen i executen en la cadena de blocs Ethereum i, també es va crear una criptomoneda: l'ether. En aquest projecte es detallarà més extens aquesta plataforma, ja que ha estat l'escollida per a la implementació de la Dapp.

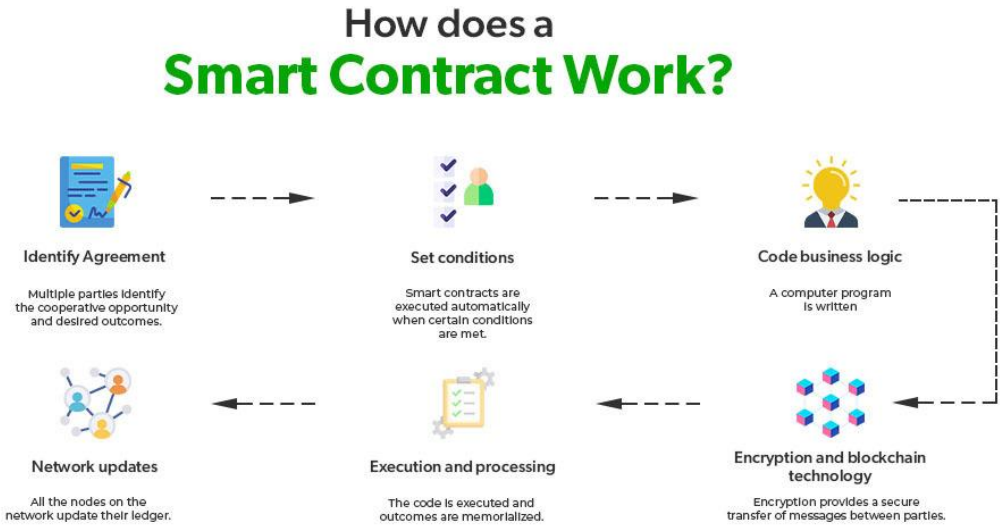


Figura 2: Imatge mostrant com funciona un smart contract. [3]

En el cas d'Ethereum, en lloc de la prova de treball que es parlava en Bitcoin ara s'utilitza la prova de participació (POS), qui valida el bloc es selecciona a l'atzar segons unes condicions que es detallaran més endavant.

En definitiva, cada vegada són més els negocis que veuen el seu potencial i que ho volen integrar en la seva activitat la qual es troba en un període d'expansió.

2.2 Definició i característiques

Blockchain o la cadena de blocs és un tipus de base de dades descentralitzada que emmagatzema la informació en forma de blocs, on en cadascun es guarda informació sobre les transaccions realitzades, el temps en el qual el bloc va ser afegit a la cadena i el hash del bloc anterior. D'aquesta manera aconseguim que la validesa de les transaccions futures puguin ser verificades consultant l'últim estat de l'adreça d'enviament.

De manera general, la tecnologia acompleix tres activitats com a part del seu funcionament:

- Valida la informació d'entrada.
- Assegura la informació d'entrada.
- Preserva el registre d'activitats.

És una tecnologia innovadora que ha revolucionat i està per fer-ho encara la indústria i la manera en què vivim. A continuació es descriuen algunes de les característiques principals:

1. **Descentralització:** és una base de dades que no està controlada per una entitat, el que significa que les dades són més segures i resistents a la manipulació.

2. **Seguretat:** tècniques criptogràfiques per a protegir les dades i les transaccions, el que fa que sigui molt difícil o quasi impossible per als atacants accedir o modificar-ho.

3. **Transparència:** totes les transaccions realitzades són transparents i es poden veure públicament, cosa que les dades sempre queden registrades.

4. **Immutabilitat:** una vegada que es registra una transacció, no es poden eliminar ni modificar. Això assegura que les dades siguin precises i de confiança.

5. **Eficiència:** permet dur a terme transaccions ràpidament sense la necessitat d'intermediaris amb un cost elevat.

6. **Anonimat:** permet fer transaccions sense revelar la identitat dels usuaris, el que és molt útil per a les transaccions financeres. Com pot ser el cas de la implementació de l'aplicació de votació només amb comptes blockchain.

Quan parlem de blockchain, un dels conceptes més interessants és la seva arquitectura distribuïda, ja que, amb ella s'elimina la centralització. Existeixen tres tipus importants d'estructures:

- **Estructura centralitzada:** tota l'estructura és gestionada per un sol node i els seus usuaris són de la mateixa comunitat. S'utilitza principalment en serveis web, per la qual cosa han de passar totes les persones per accedir a elles (Facebook, Wikipedia)

- **Estructura descentralitzada:** està dividida en diversos nodes operatius que funcionen com la seva pròpia estructura centralitzada, fragmentant el servidor central en petits servidors distribuïts. Qualsevol usuari de l'estructura pot accedir a les dades d'altres independentment al node al qual pertanyen (GNU Social, Diaspora).

- **Estructura p2p (peer to peer):** és una estructura on els nodes participen de forma completament descentralitzada. És a dir, és una xarxa on no hi ha un punt central de connexió o control, i on les parts actuen de manera autònoma. D'aquesta manera, els integrants de la xarxa poden intercanviar informació de manera directa i sense intermediaris

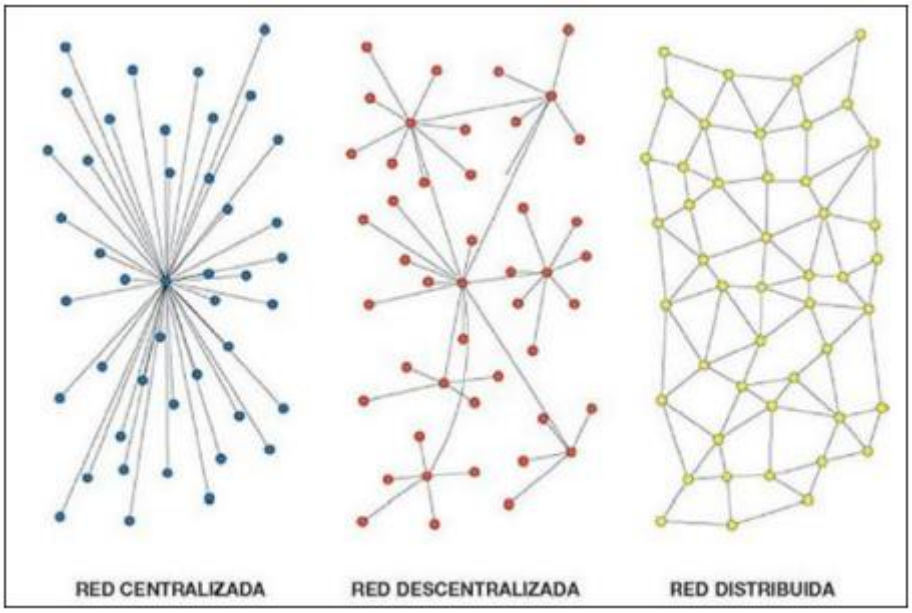


Figura 3: Imatge demostrant la diferència entre les estructures. [5]

Un dels exemples més importants d'una arquitectura distribuïda (P2P) és el de Bitcoin, en el qual tots els usuaris veuen totes les transaccions i tenen els mateixos drets. Ethereum utilitza l'arquitectura P2P, igual que Bitcoin, aconseguint així que tot el programari desenvolupat en aquest llenguatge sigui totalment lliure i descentralitzat, ja que tots els usuaris de la xarxa tenen accés a tots els contractes de la cadena de blocs i al codi font d'aquests.

2.3 Tipus de blockchain

Existeixen diversos tipus de blockchain, cadascuna amb les seves capacitats i característiques que s'adapten a diferents necessitats. Aquestes són: la publica, la priva i l'híbrida.

- **Blockchain publica:** és el primer tipus que va existir, i es refereix a les cadenes de blocs a les que qualsevol persona pot accedir i unir-se. Aquest tipus de blockchain manté obert al públic les seves dades, software i desenvolupament.

  Entre les característiques de la blockchain pública permeten que qualsevol persona formi part d'aquesta, sigui com a usuari, miner o administrador d'un node. El funcionament de la xarxa és completament transparent i obert, a més que no existeix entitats centralitzades cosa que significa que no hi ha autoritat que reguli el funcionament.

  Per últim, el manteniment econòmic de la blockchain depèn del sistema integrat en aquesta, ja que generalment el sistema depèn de la mineria i el cobrament de comissions per cada transacció que es realitzi dins de la xarxa.

- **Blockchain privada:** amb l'evolució de la tecnologia blockchain i la seva expansió, moltes empreses van començar a interessar-se en ella. Això va derivar en el desenvolupament de solucions blockchain privades.

  Aquest tipus de blockchain generalment compta amb els mateixos elements que una blockchain pública, però a diferència d'aquestes, les blockchain privades depenen d'una unitat central que controla totes les accions dins d'aquesta.

  L'accés a la xarxa està restringit a elements que sol poden ser autoritzats per la unitat central de control, com per exemple per tenir accés al llibre de transaccions. El manteniment econòmic de la blockchain depèn de l'empresa responsable del projecte. No s'acostuma a treballar amb criptomonedes ni es realitzen accions de mineria.

- **Blockchain híbrida:** Aquest tipus de blockchain és una fusió entre les blockchain públiques i les privades. És un intent d'aprofitar el millor de tots dos mons. En aquestes blockchain, la participació en la xarxa és privada. És a dir, l'accés als recursos de la xarxa és controlat per una o diverses entitats. No obstant això, el llibre de comptabilitat és accessible de manera pública.

  Això significa que qualsevol persona pot explorar bloc a bloc tot el que succeeix en aquesta blockchain.
  Un perfecte cas d'ús està succeint en el sector sanitari, on es comença a usar blockchain per a emmagatzemar les dades de les seves línies de producció de medicaments.

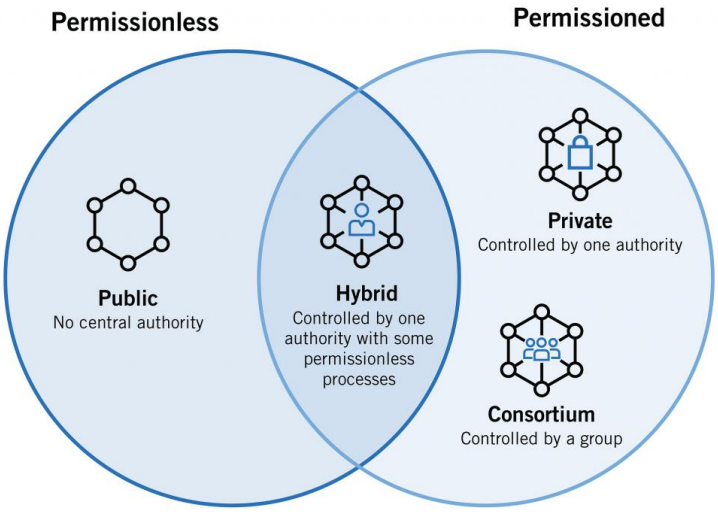


Figura 4: Imatge representant els tipus de blockchain. [7]

2.4 Algorismes de consens

Un algorisme de consens pot ser definit com el mecanisme a través del qual una xarxa Blockchain aconsegueix un consens. Com s'ha explicat anteriorment, les cadenes de blocs necessites assolir entre els nodes que la mantenen, un consens per assegurar la seguretat de la xarxa, validant les transaccions que s'estan realitzant.

Existeixen molts algorismes per a fer-ho possible, per això a continuació s'explicarà els més comuns que existeixen en el mercat criptogràfic.

- **Proof-of-worf (PoW):** és el primer algorisme de consens de la blockchain. Els miners solucionen operacions matemàtiques completes que requereixen potència computacional. En el PoW cada node calcula constantment el valor hash, el consens requereix que el valor calculat sigui igual o menor que un determinat valor hash donat.

  Els miners han de confirmar l'exactitud del valor que un node obtingui, després d'això les transaccions seran validades i aprovades que es denota mitjançant un nou bloc en la cadena. Al miner que aconsegueixi el valor hash, rep un incentiu de la moneda com a recompensa.

- **Proof-of-Stake (PoS):** aquest algorisme neix com a alternativa del PoW i té com a objectiu un consens distribuït. El seu funcionament consta que els usuaris demostrin la quantitat de monedes que posseeixen. Reemplaça la mineria intensiva per un mecanisme on els blocs es validen segons a la participació dels involucrats.

  Aquest algorisme és més efectiu i amb més estalvi d'energia, algunes plataformes demanen el canvi de PoW a PoS per aquest principal motiu, no obstant aquest és més propens a atacs a la xarxa.

- **Prova de participació delegada (DPOS):** és molt ràpid i més coneguda per la seva implementació en EOS (S.O per a aplicacions blockchain), i a vegades se'l coneix com a democràcia digital. La forma en què funciona és que els usuaris voten per "delegats" a qui se'ls hi dona poder d'obtenir beneficis a l'executar un node complet. El pes del seu vot depèn de la seva participació o bloqueig de monedes.

  Els delegats desitgen rebre la quantitat més grans de vots possibles, se'ls incentiva a crear coses valuoses per a la comunitat, ja que probablement reben vots addicionals.

- **Tolerància Bizantina a falla delegada (dBFT):** és un algorisme que inclou una màquina d'estat amb tècniques de replicació, amb la finalitat de tolerar errors bizantins fins a un terç de tota la xarxa. En aquest mecanisme és triat un node en cada ronda segons unes normes i uns vots dels nodes a favor seu; aquest node escollit és responsable d'establir la transacció. Per tant, requereix que tots els nodes siguin reconeguts per la xarxa.

- **Ripple:** s'utilitza en subxarxes, el qual pot ser aplicat dins d'una xarxa més gran. Pot tenir dos tipus de node: un node servidor participant del procés de consens i un node client per a transferir els fons. Així, cada node servidor té una llista de nodes únics (UNL), els quals són consultats per a realitzar la validació de transaccions i col·locar-les al registre blockchain, si els acords rebuts arriben al 80%, la transacció s'emmagatzemarà.

2.5 Criptografia i seguretat

La criptografia és la columna vertebral darrere la seguretat de la blockchain. La podem definir com un camp d'estudi el qual l'objectiu és desenvolupar comunicacions privades segures en presència d'atacants.

El context de la criptografia en la blockchain es fa servir per a diverses finalitats: dur a terme un intercanvi de dades entre dos nodes que realitzen transaccions, verificar que aquestes transaccions a la xarxa mitjançant nodes miners, garantint la immutabilitat dels registres i la seva confidencialitat de les dades en els contractes intel·ligents.

Existeixen diferents tipus de xifratges dins de la criptografia, on cadascun té diferents funcionalitats. A continuació es detallarà alguns sistemes actuals:

- **Xifratge simètric o de clau privada:** fa ús de la mateixa clau per a xifrar que per a desxifrar el contingut entre les dues parts, per tant, ambdós s'han de posar d'acord i decidir la clau a utilitzar. El seu avantatge és l'alta velocitat d'execució.

  Encara que, el xifratge simètric té una greu limitació: si la clau arriba a ser coneguda per a persones malintencionades, tota la comunicació es veu compromesa. Per aquesta raó és poc pràctic per a una tecnologia descentralitzada.

- **Xifratge asimètric o de clau pública:** s'utilitzen un parell de claus, una pública i una privada per assegurar la comunicació. L'emissor usa la clau pública per a xifrar el missatge, mentre que el receptor fa servir la clau privada per a desxifrar-lo. Ambdues claus pertanyen al receptor de la comunicació. Cal destacar que la clau pública la pot tenir qualsevol però la privada només el propietari del missatge.

  És extremadament difícil que algú sigui capaç de desxifrar el missatge, ja que sol l'emissor i el receptor coneixen les dues claus. A més, es generen només una vegada pel que és impossible que es generin dos parells de claus iguals.

- **Hash:** considerat un xifratge molt segur, no usa claus. Fa servir un algorisme per aprendre entrada de qualsevol longitud i produir una sortida de longitud fixa. Les funcions hash produeixen col·lisions baixes, cosa que les possibilitats que dues entrades condueixin a la mateixa sortida són astronòmicament petites fent impossibles els atacs de força bruta.

  El missatge original no pot ser objecte d'enginyeria inversa a partir del resultat del hash. Són deterministes, és a dir, qualsevol entrada produirà la mateixa sortida quan s'executa a través de la funció hash, inclús un petit canvi en l'entrada, com canviar un bit, produirà un hash totalment diferent.

  Per exemple, a Bitcoin es fa servir la funció SHA-256, mentre que a Ethereum es basa en KECCAK-256.

  - **SHA-256:** funció hash que produeix un valor de 256 bits de longitud. Modera la creació i gestió d'adreces i verifica les transaccions. Bitcoin utilitza el doble SHA-256, és a dir, que aplica dues vegades les funcions hash.

    Es publica l'any 2001 junt amb el conjunt de funcions SHA-2, dissenyades per l'Agència de Seguretat Nacional. És extremadament segur i el seu funcionament no es coneix en el domini públic. El govern dels Estats Units el fa servir per protegir informació delicada, gràcies a la seva capacitat de verificar un contingut de dades sense revelar-lo a causa de l'ús de signatures digitals.

    De fet, és gairebé impossible revelar les dades inicials del mateix valor hash. A més, a atac de força bruta és molt poc probable que triomfi gràcies al nombre astronòmic de combinacions potencials.

- **KECCAK-256:** Ethereum addicionalment fa servir aquesta funció criptogràfica la qual agafa un contingut i el converteix en una cadena de 32 bytes o 256 bits, aquesta funciona de la mateixa manera que SHA-256.

Encara que tant SHA-256 com Keccak-256 són algorismes criptogràfics que produeixen sortides de hash de 256 bits, difereixen en l'algorisme subjacent utilitzat per a la generació de hash, la qual cosa resulta en diferències en la seguretat i resistència a les col·lisions.

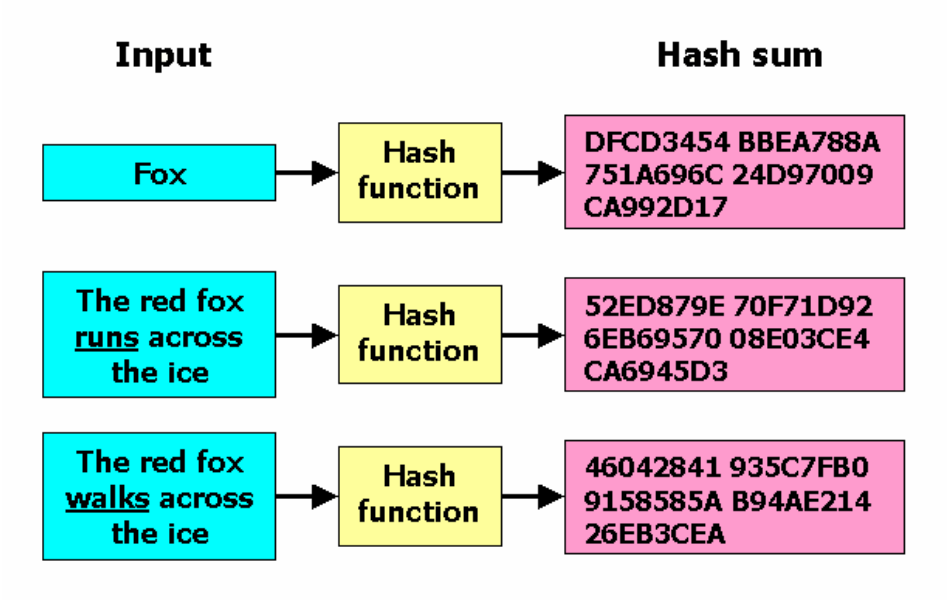


Figura 5: Exemple de funció hash. [12]

3. Ethereum

3.1 Origen i característiques

Ethereum és una plataforma blockchain descentralitzada que permet la creació d'aplicacions descentralitzades i contractes intel·ligents. Va ser proposada el 2013 pel programador canadenc Vitalik Buterin, qui va presentar per primer cop la idea en un document tècnic (whitepaper) publicat el novembre d'aquell any.

Buterin, que havia estat un desenvolupador actiu de Bitcoin, es va adonar que la funcionalitat de Bitcoin era limitada i volia crear una plataforma que permetés la creació d'aplicacions descentralitzades i contractes intel·ligents. Per això, Buterin i un equip de desenvolupadors van començar a treballar en la creació d'una nova plataforma blockchain que permetés l'execució de programes complexos a la xarxa blockchain.

El juliol de 2014, l'equip de desenvolupament d'Ethereum va llançar un fons de recaptació de fons, anomenat "Ether Sale", per finançar el desenvolupament de la plataforma. El fons va recaptar 31,5 milions de dòlars en ether, la criptomoneda nativa de la xarxa Ethereum.

El juliol de 2015, l'equip va llançar la primera versió d'Ethereum, anomenada "Frontier". Des de llavors, Ethereum ha evolucionat significativament, amb diverses actualitzacions importants de la xarxa.

Avui en dia, Ethereum és una de les plataformes blockchain més populars i utilitzades al món, amb milers d'aplicacions descentralitzades i contractes intel·ligents que s'executen a la seva xarxa. A més del seu ús en l'àmbit de les criptomonedes, Ethereum fa servir en una varietat d'aplicacions, des de finances descentralitzades i jocs en línia fins a sistemes d'identitat i emmagatzematge de dades descentralitzats.

La plataforma Ethereum proporciona una moneda d'intercanvi per a realitzar transaccions anomenades Ether, també pot ser usat per a incentiu als miners de la plataforma. A més, la plataforma proporciona una màquina virtual (EVM) on es poden executar els contractes intel·ligents fent servir una xarxa de nodes. El nombre d'operacions computacionals està limitat a causa del gas, concepte que s'explicarà més tard.

Aquests són uns dels elements principals de la plataforma que es detallaran en els següents punts.

3.1.1 Màquina Virtual de Ethereum (EVM)

És una màquina virtual que forma part de l'ecosistema d'Ethereum. La seva funció és la de permetre l'execució de programes o smart contracts amb la finalitat de desplegar sobre la blockchain una sèrie de funcionalitats.

Per a fer més senzilla la programació per a aquesta màquina virtual es va crear el llenguatge d'alt nivell anomenat Solidity. A través d'aquest llenguatge es facilita la creació dels smarts contracts.

En primer lloc, es transforma Solidity als codis d'operació (OP_CODES) i després a un bytecode. Aquest bytecode és finalment executat per l'EVM per a efectuar les operacions especificades en un smart contract. Tot això fa que l'EVM pot funcionar com un computador de veritat, executant des de les més senzilles fins a les més complexes operacions.

Funcionant com una màquina de pila que empeny valors transitoris fins i des d'una pila pushdown, l'EVM té una profunditat de 1024 elements, sent cadascun d'ells una paraula de 256 bits. També té una memòria temporal en forma de matriu de bytes que canvia entre dues transaccions en la blockchain d'Ethereum.

El millor de tot és que per a poder usar el potencial d'EVM tan sols hem de tenir alguna cosa d'ether i interactuar amb alguna DApp, contracte intel·ligent o fer el nostre propi contracte. No existeix cap limitació, qualsevol pot aprofitar el poder que EVM té a la seva disposició.

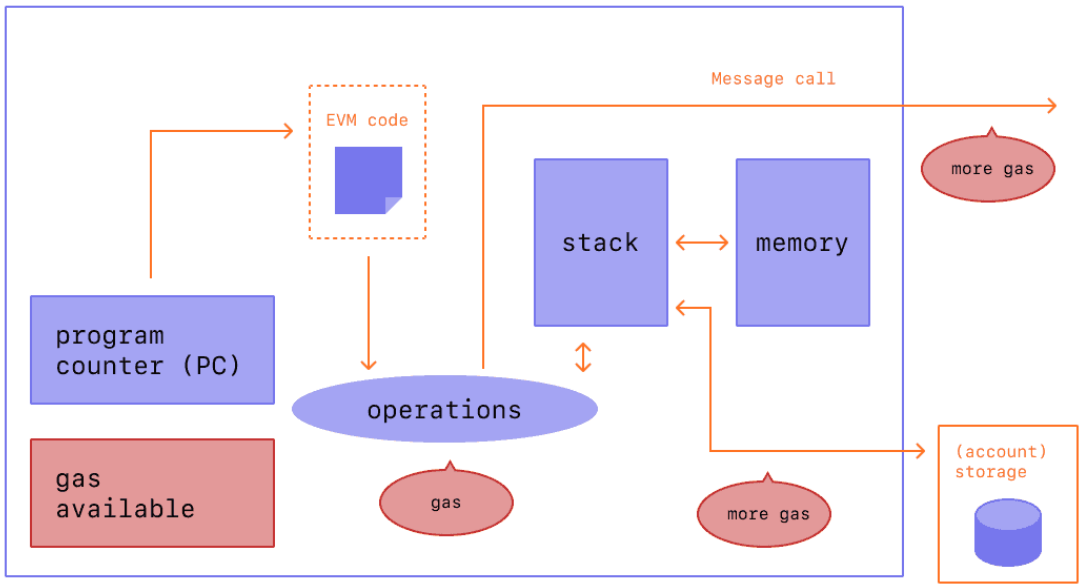


Figura 6: Arquitectura de la EVM. [18]

Alguns dels avantatges que ens proporciona és executar codi sense preocupar-se del seu impacte en la resta de la xarxa o que hi hagi problemes amb dades en qualsevol dels ordinadors dels nodes. A més, poder executar contractes intel·ligents en diferents entorns informàtics. Això permet que el fallo d'un sol node no tingui cap impacte negatiu en l'execució de la dApp.

Per altra banda, en els inconvenients, tenim les elevades comissions per transacció o costs de gas associats a l'execució d'un contracte intel·ligent, i l'actualització posterior dels contractes, porta un risc de seguretat associat a la creació d'un contracte intel·ligent entremig que faci referència a la direcció del contracte original.

Encara que els canvis revolucionaris que ha suposat l'EVM en l'ecosistema de la cadena de blocs, aquesta tecnologia de lectura i execució de codi està sent millorada. El següent objectiu és passar de l'EVM a Ethereum WebAssembly (eWASM). Dissenyat per a ser altament modular i independent de la plataforma.

3.1.2 Ether

Ether, el token natiu de la blockchain Ethereum, té un caràcter dual: moneda financera com d'utilitat. Per una banda, és la criptomoneda amb la qual es paguen els serveis computacionals d'aquesta xarxa i les tarifes de verificació de les transaccions. Per l'altra, és la base principal que es permet que s'executin aplicacions descentralitzades, aplicacions que els usuaris duguin a terme transaccions i acords directes sense intermediaris.

El valor de l'ether es determina pel mercat, com qualsevol altre actiu financer, i es pot cotitzar en diferents intercanvis de criptomonedes. A més, l'ether també pot ser minat, tot i que el procés de mineria és diferent del que s'utilitza per a altres criptomonedes com el bitcoin.

Avui en dia hi ha un valor de mercat de 220 milions de dòlars en ether, amb una quantitat circulant de 120 milions d'ethers. Sent així la segona moneda més gran del mercat quant a capitalització del mercat.

En l'actualitat, cada bloc nou creat a la xarxa Ethereum recompensa el miner amb 2 ethers. Això pot canviar en el futur, ja que l'Ethereum està en constant evolució i els seus protocols poden ser actualitzats per a modificar aquesta i altres característiques de la xarxa.

3.1.3 Gas

El Gas en Ethereum és una unitat utilitzada per a mesurar el treball fet per Ethereum per a realitzar transaccions o qualsevol interacció dins de la xarxa.

Perquè puguis dur a terme una transacció que sigui acceptada i inclosa en la blockchain, s'ha de pagar una comissió perquè els miners acceptin la teva transacció i la inclogui en un bloc. Una vegada allí, aquesta transacció serà validada per la xarxa, i llavors, s'executaran les ordres que estan en la transacció, només així podràs dir que ha estat acceptada i confirmada.

A Ethereum els desenvolupadors van decidir assignar valors constants a les diferents operacions que es poden portar a cap en Ethereum. D'aquesta manera, cada tasca en Ethereum té un valor de Gas estipulat, que no canvia i no és alterat per la pujada o baixada en el valor de l'ether, la moneda nativa d'Ethereum.

L'anterior genera tres coses que són importants i vitals dins d'Ethereum, i que expliquem a continuació:

- **Unitat de Gas:** La Unitat de Gas és la quantitat de Gas que es pot atribuir a una instrucció en específic.

- **Preu de Gas:** El Preu de Gas per la seva part és el pagament de comissió que fem per cada Unitat de Gas. És un preu que es tria pagar per cada unitat i el fem usant unitats decimals d'ether, els anomenats Gwei. Aquesta comissió és la que et permet tenir prioritat d'atenció. Si pagues més per cada Unitat de Gas que facis servir, més ràpid els miners acceptaran la teva transacció i la portaran a un bloc.

Denominations of Ether		
Unit Name	Wei Value	Number of Wei
Wei (wei)	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mwei (lovelace)	1e6 wei	1,000,000
Gwei (shannon)	1e9 wei	1,000,000,000
Twei (szabo)	1e12 wei	1,000,000,000,000
Pwei (finney)	1e15 wei	1,000,000,000,000,000
Ether (buterin)	1e18 wei	1,000,000,000,000,000,000

Investopedia

Figura 7: Denominacions de ether. [20]

- **Límit de Gas:** Aquest és un valor que indica la quantitat màxima d'Unitats de Gas que la xarxa Ethereum pot manejar en un moment donat. És el seu límit màxim, i és un punt que els miners no poden sobrepassar en cap moment.

Per a calcular el preu en ether d'una transacció, simplement has de multiplicar el nombre de Gwei de la teva transacció pel nombre de GAS. Per tant, si Gwei=52 i GAS = 21000.

Transacció → 52 * 21000 = 1092000 Gwei = 0.001092 ether

El preu de ether actualment és de 1806.50 $, com a resultat, la transacció costaria uns 1.97 $.

3.2 Funcionament de les transaccions

A continuació s'explicarà pas a pas com funciona una transacció Ethereum, ho podem definir com un procés el qual un usuari envia ether d'un compte a un altre dins de la xarxa.

Cada usuari té una direcció única que identifica el seu compte en la xarxa. Aquesta direcció es genera a partir d'una clau pública derivada d'una clau privada única que només l'usuari ha de conèixer. La clau privada és utilitzada per a signar transaccions i garantir l'autenticitat d'aquestes.

El remitent de la transacció crea un objecte de transacció que conté informació com l'adreça del destinatari, la quantitat d'ether a transferir i qualsevol dada addicional necessària per a l'execució de contractes intel·ligents. El mateix remitent firma la transacció fent servir la seva clau privada. Això ho aconsegueix aplicant una funció de firma criptogràfica a la transacció (KECCAK-256). La firma és única i verifica que el remitent sigui el propietari del compte.

Una vegada signada, la transacció es propaga a través de la xarxa Ethereum. Els nodes de la xarxa la reben i la validen, assegurant-se que la signatura sigui vàlida i que el remitent tingui suficients fons per a realitzar la transacció. Els nodes competeixen per a resoldre el Proof-of-Work, el primer a assolir-ho rep les recompenses i el bloc es considera confirmat i és afegit a la cadena de blocs.

Un cop que la transacció està confirmada, els saldos dels comptes s'actualitzen reduint el del remitent i incrementant el del destinatari.

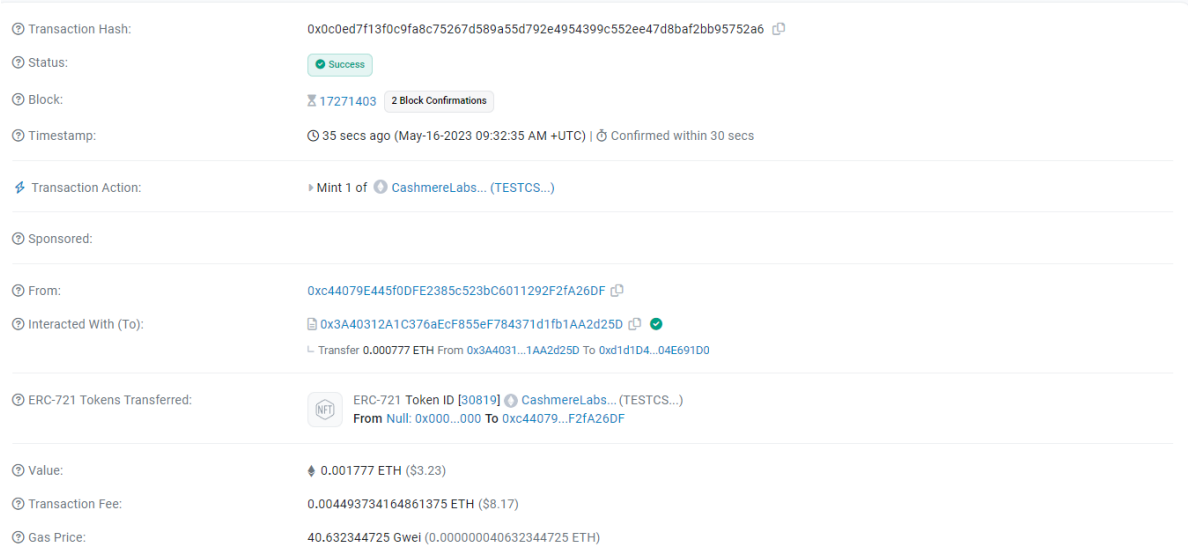


Figura 8: Estructura d'una transacció. [25]

3.3 Ethereum 2.0

La xarxa Ethereum estava experimentant colls d'ampolla simplement per la quantitat d'activitat en la blockchain. Per exemple, les taxes de gas pagades als miners pel seu treball aconseguien a vegades nivells extraordinàriament alts. La xarxa estava estancada per limitacions tècniques, en concret la congestió de la xarxa, l'escalabilitat i l'accessibilitat.

És per aquesta raó que es va començar a parlar sobre una actualització que resolgués tots els problemes de la xarxa: Ethereum 2.0. Les millores en aquestes àrees eren i continuen sent fonamentals. Ethereum és la cadena de blocs en la qual s'allotgen moltes aplicacions descentralitzades basades en contractes intel·ligents (dApps), que tenen aplicacions en finances, béns, cadenes de subministrament, entre moltes altres.

Però per a tenir l'escalabilitat prevista en tots els sectors i usos, la cadena de blocs havia de ser capaç de gestionar interaccions de xarxa a una escala molt major.

Un dels canvis realitzats i el més important, és l'algorisme de consens. El Proof-of-Stake és més ràpid i ecològic que el Proof-of-Work, ja que consumeix molta menys energia. Això es deu al fet que PoS no és una competició per a veure quin miner pot arribar abans a la solució del hash del bloc, que és el que requeria tanta energia. En el seu lloc, els protocols de xarxa seleccionen aleatòriament els nodes que validen les transaccions i obren nous blocs.

Un altre canvi és la fragmentació de la blockchain en cadenes més petites. Aquestes funcionaran de manera independent processant les seves pròpies transaccions, de manera que proporciona més escalabilitat i capacitat de processament de transaccions per segons.

S'ha de tenir en compte que el seu desenvolupament no tingui cap impacte amb els smarts contracts, tokens i altres elements creats a la blockchain d'Ethereum. Aquests ajustaments tindran un impacte positiu en les dApps, que passaran a tenir una millora en el seu rendiment.

3.4 Diferències entre Ethereum i Bitcoin

Donar a conèixer les principals diferències entre Bitcoin i Ethereum, dos dels majors projectes blockchain.

Comencem parlant del nivell de descentralització, Bitcoin és la criptomoneda més descentralitzada que existeix en el món, amb la xarxa amb la quantitat més gran de nodes, miners, desenvolupadors, etc.

Per altra banda, Ethereum és un projecte que en el seu moment es va veure afectat per trencar la immutabilitat de la blockchain, al reescriure un part del seu historial per a esborrar el robatori en "The DAO" (Organització Autònoma Descentralitzada) i recuperar els fons de l'organització. Cosa que va generar una forta divisió en la comunitat i va donar origen a Ethereum Classic, que seria la criptomoneda original d'Ethereum.

Una altra de les diferències és l'emissió de criptomonedes. En el cas de Bitcoin hi ha un total de 21 milions de bitcoins, mai es podrà superar aquesta quantitat. A més aquesta emissió va disminuint amb el temps fins a arribar a 0. Això és degut al fet que la xarxa fa servir operadors de desplaçament de bits, és a dir, operadors matemàtics que arrodoneixen alguns punts decimals al número enter més petit.

Aquest arrodoniment pot ocórrer quan la recompensa del bloc per produir un nou bloc de bitcoin es divideix per la meitat, i es calcula la quantitat de la nova recompensa. Aquesta, es pot expressar en satoshis, sent un satoshi equivalent a 0,00000001 bitcoins.

Ethereum, en canvi, té una emissió infinita de monedes, on en l'actualitat existeixen més de 120 milions d'ether en circulació i continuaran generant-se més.

La mineria és una altra de les grans diferències entre Bitcoin i Ethereum. En primer lloc, Bitcoin usa el conegut model de Prova de Treball (Proof of Work – PoW), usant l'algorisme HashCash i la funció hash SHA-256 per a fer el treball computacional.

Una altra de les característiques importants de la mineria en Bitcoin és que aquesta genera un nou bloc aproximadament cada 10 minuts, pateix d'ajustos de dificultat cada 2016 bloc (uns 14 dies) i té un halving (divisió a la meitat de la recompensa de bloc) cada 210.000 blocs (uns 4 anys aproximadament).

Per part seva, Ethereum fa servir, des de setembre de 2022 amb The Merge, l'anomenada Proof of Stake (Post), o Prova de Validació, la qual cosa ha reduït fins a un 99% la contaminació que generava el fet de realitzar les transaccions en la blockchain d'Ethereum.

La mineria en Ethereum generava un nou bloc aproximadament cada 10-20 segons, sofria d'ajustos de dificultat de manera contínua, i no tenia un sistema de halving pròpiament dit, sinó que el seu valor d'emissió disminuïa d'acord amb un consens aconseguit en la comunitat.

Per últim punt a tractar, es parlarà de la gran diferència que hi ha amb l'escalabilitat. En Bitcoin, està limitada actualment a unes 7-8 transaccions per segon. Però en Ethereum els valors arriben fins a 16-20 transaccions per segon. Duplica els primers valors, però sol presentar majors nivells de cogestió que Bitcoin, ja que compta amb l'activitat de milers de tokens addicionals que congestionen la xarxa.

	Bitcoin	**Ethereum**
Descentralització	Alta	Mitja
Preu moneda (18/05/23)	27413,39$	1827,55$
Emissió	21.000.000	Il·limitada
Mineria	PoW	PoS
Escalabilitat	7-8 transaccions per segon	16-20 transaccions per segon

Taula 1: Comparació entre Bitcoin i Ethereum.

4. Smart contracts

4.1 Què són?

Un contracte és un acord entre dues o més parts, un entorn on es defineix una sèrie de condicions del que es pot fer, com es pot fer, que passa si no es fa... És a dir, unes normes que permeten a les dues parts que ho accepten entendre en què consistirà la interacció.

Un contracte intel·ligent és capaç d'executar-se ell mateix, de forma autònoma i automàtica sense intermediaris. Es tracta de scripts escrits amb llenguatges de programació, això vol dir que les condicions del contracte són sentències i comandes en el codi que el forma. És un codi que és visible per a tots i que no es pot canviar existir sobre la tecnologia blockchain.

És important destacar que, en estar distribuït per milers d'ordinadors, s'evita que una gran companyia els custodiï, la qual cosa elimina burocràcia, censures i els grans costos i temps implícits d'aquest procés.

La idea de smart contract va ser exposada per Nick Szabo, qui va exposar que, mitjançant el desenvolupament d'aplicacions informàtiques amb les quals mitjançant l'ús de claus criptogràfiques segures, es podia executar de manera automàtica una sèrie d'accions (pagament d'un preu, lliurament de la cosa objecte d'un contracte, en sentit estricte, etc.).

Així, per exemple, es podria contractar la compravenda d'un cotxe mitjançant un contracte digital o manual. A través del smart contract, la transferència de diners podria quedar pendent a l'espera que, també en forma digital, es registri el lliurament de la clau. Una vegada executada aquesta acció el programa, en forma automàtica, allibera els fons i s'acaba per concretar-se la transferència del preu.

Precisament, perquè els contractes intel·ligents es puguin executar, és necessari que existeixen les transaccions programables i un sistema financer que les reconegui. Aquí és quan en 2009 quasi 15 anys després es faria realitat amb l'aparició de la cadena de blocs.

Com bé s'ha explicat anteriorment Ethereum és un dels projectes més famosos en el sector de smarts contracts. Una plataforma de computació distribuïda basada en blockchain que permet executar els contractes intel·ligents en una màquina virtual descentralitzada (Ethereum Virtual Machine EVM).

Ethereum ha creat un intèrpret de llenguatge de programació molt més extens (Turing complet), permetent afegir lògica molt més complexa dins del blockchain. És a dir, es podria assemblar a un ordinador distribuït, el qual utilitza la seva

criptomoneda (ether) com la “gasolina” que necessiten el contracte perquè els miners puguin executar-lo. És a dir, ara els contractes són programes amb moltes més funcionalitats i possibilitats.

Una màquina de Turing universal és un model teòric de computadora que pot simular altres màquines de Turing i resoldre qualsevol problema computacional que pugui ser resolt algorítmicament. Això significa que pot realitzar càlculs i executar programes de qualsevol complexitat.

Ethereum, fa servir un llenguatge de programació específic anomenat Solidity. Solidity és un llenguatge Turing complet, el que significa que pot expressar tota la classe de problemes computacionals resolubles en una màquina de Turing universal.

Això és important perquè permet als desenvolupadors de smart contracts implementar lògica empresarial complexa i algoritmes sofisticats dins dels seus contractes intel·ligents. Amb Solidity, es pot crear funcions, bucles, condicionals i altres elements clau d'un llenguatge de programació per expressar càlculs i comportaments sofisticats.

No obstant això, cal tenir en compte que, tot i que els smart contracts són Turing complets, hi ha restriccions en el seu funcionament per garantir la seguretat i l'eficiència de la plataforma blockchain. Això implica imposar límits en el temps d'execució, la complexitat del càlcul i altres aspectes per evitar possibles abusos i atacs maliciosos.

A continuació es comentarà una llista de beneficis dels smarts contracts que aclareixen el perquè s’haurien de fer servir:

- **Lliure d’interrupcions:** la part més molesta és quan una tercera persona o un tercer interromp l’acord. Amb els smart contracts no hi haurà interrupció per part de cap tercera persona. Només tu prendràs totes les decisions i faràs qualsevol transacció tu mateix.
  No has d'esperar la confirmació de cap advocat o corredor. Per tant, la totalitat del procés estarà completament lliure de manipulació.

- **Seguretat:** Avui dia els atacs informàtics s'han convertit en un problema. El smart contract oferirà un lloc web per als teus documents lliure d'atacs informàtics. S'encripten aquests llocs web amb la codificació de més alt nivell, que els fa gairebé impossibles de desxifrar.

- **Execució ràpida:** Processen cada tasca automàticament amb codis de programari i internet. Això estalviarà molt temps. A més, no hi haurà processament manual.

- **Sense errors:** Hi haurà un registre complet amb cada detall de tots els termes i condicions. Per tant, el processament estarà lliure d'errors, de manera més ràpida i econòmica.

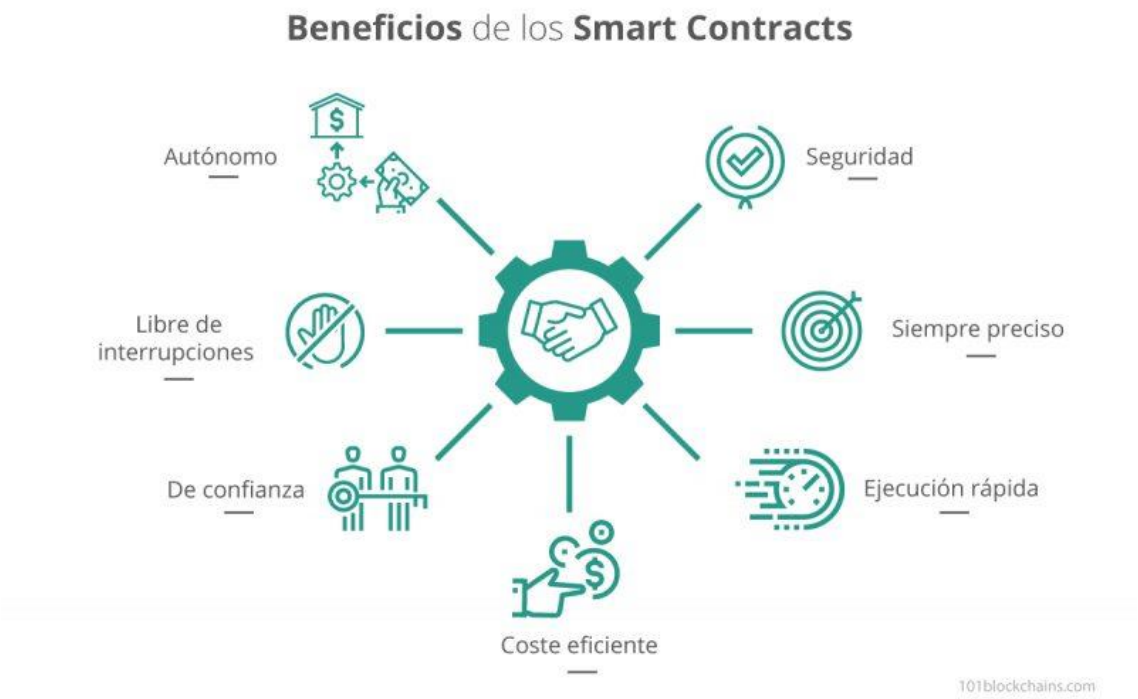


Figura 9: Beneficis dels contractes intel·ligents. [28]

4.2 Funcionament dels contractes

Per a entendre millor el seu funcionament el millor és col·locar un exemple popular. Imaginem una màquina expenedora de menjar d'aquestes que podem trobar fàcilment en alguns aeroports.

Aquesta màquina es troba programada perquè després d'introduir una certa quantitat de diners i un codi, puguis obtenir el menjar o la beguda que has seleccionat.

Però, també aquesta màquina està programada per a retornar-nos diners en cas que hàgim introduït de més i també per a avisar-nos si el producte seleccionat no es troba disponible.

En aquest cas, la configuració d'aquesta màquina seria el contracte intel·ligent i les parts implicades seríem la màquina i nosaltres. Les condicions d'aquest contracte serien: donar-te menjar, donar-te diners, avisar-te de la disponibilitat d'un producte, etcètera.

Si analitzem el seu funcionament des d'un aspecte més tècnic, és important recalcar que el mateix és similar al de qualsevol altra transferència sobre blockchain, però la idea del smart contract es pot descriure amb els següents passos:

1. **Identificar les parts i establir condicions de l'acord:** identificar les parts implicades i les condicions del contracte. Es descriuen les obligacions de cada part i les normes d'execució del contracte

2. **Definir les condicions d'execució del contracte:** el segon pas consisteix a especificar les condicions que s'han de complir perquè s'executi el contracte. Aquestes condicions solen ser un conjunt de normes o criteris.

3. **Escriure el codi del contracte intel·ligent:** el codi especificarà els passos exactes que s'han de donar per a executar el contracte quan es compleixin les condicions especificades. El llenguatge de programació més popular és Solidity que com s'ha comentat abans segueix el Turing complet. Però trobem altres com: Vyper, Serpent, LLL.

4. **Desplegar el contracte en una plataforma blockchain:** es tracta de validar el contracte pujant el codi a la xarxa blockchain. Això implica enviar una transacció especial anomenada "transacció de desplegament" que inclou el bytecode del smart contract. El desplegament del contracte implica el pagament d'una tarifa de desplegament, coneguda com a "gas", que compensa els miners per executar el codi del contracte.

5. **Activar l'execució del contracte:** quan es compleixen les circumstàncies predeterminades, el contracte s'executa automàticament i la xarxa blockchain l'activa. Això pot incloure crides a les funcions definides al contracte, que són executades per l'EVM. Les funcions poden prendre paràmetres i retornar resultats segons la lògica programada.

6. **Registrar els detalls del contracte a la blockchain:** la informació del contracte s'introdueix a la xarxa de la blockchain. Inclou les condicions del contracte, els requisits previs, la data i l'hora de l'execució. Les dades del contracte són immutables un cop s'introdueixen a la cadena de blocs, per tant, no es pot modificar.

La seguretat dels smart contracts és un aspecte crucial. Cal tenir en compte possibles vulnerabilitats i riscos com ara errors de programació. És important realitzar proves, auditories i revisions de seguretat per assegurar-se que el contracte funcioni com es pretén i no exposi els usuaris a riscos innecessaris.

S'està fent evident que els contractes intel·ligents han començat a reemplaçar als intermediaris tant a les empreses com a les persones físiques i que el potencial que tenen per al desenvolupament d'aplicacions descentralitzades és enorme.

Si els contractes intel·ligents compleixen el seu propòsit es pot arribar a pensar que potser algun dia vivim en un món lliure d'intermediaris.

	Contracte	Smart contract
Llenguatge	Humà	Llenguatge ordinador
Automatització	Totes les parts de l'acord	Només les transaccions que s'han d'automatitzar
Guardar	Escrites en un paper per totes les parts	Introduïda a la blockchain
Modificació	Si	No

Taula 2: Diferències entre un contracte humà i un smart contract.

4.3 Exemples de casos d'us

Hi ha diverses implementacions potencials de contractes intel·ligents que poden automatitzar el món i fer que sigui un lloc més fàcil per a viure. Aquí alguns exemples destacats de casos d'ús de contractes intel·ligents.

- **Identitat digital:** a Internet, la informació és moneda de canvi. Les empreses es beneficien de conèixer els interessos de tot el món, i les persones no sempre controlen com s'adquireixen aquestes dades, ni es beneficien d'ells. Amb els contractes intel·ligents, les persones tenen el control.

  En un futur basat en blockchain, les identitats seran "tokenitzades". Idealment, això significaria que la identitat de cada persona existeix en una cadena de blocs, fora de perill i segura. Ara, si un usuari vol participar en les xarxes socials o presentar documents a un banc per a obtenir un préstec, pot beneficiar-se del primer i controlar el procés de transacció en el segon.

  Quan es tracta de tractar amb bancs i altres institucions financeres. La comunicació només implica l'enviament de documents requerits i informació vital. No hi ha risc que un grup de préstecs emmagatzemi la teva adreça de correu electrònic i la vengui a altres empreses de crèdit. Aquesta informació està totalment sota el control de l'usuari.

- **Immobiliàries:** Tenint en compte que el procés de venda d'una casa és llarg i complicat, els propietaris contracten un agent perquè s'encarregui de les parts més confuses, com la preparació de papers i la cerca d'un comprador. Encara que això sona ideal per al venedor, els agents s'emporten una comissió significativa del preu de venda de la casa.

  Un contracte intel·ligent pot ocupar el lloc d'un intermediari, agilitzant el procés de traspàs de la casa i garantint que sigui igual d'assegurança que amb un intermediari. Imagina que l'escriptura de la teva casa està "tokenitzada" en la blockchain d'Ethereum. Si estàs llest per a vendre-la, crearies un contracte intel·ligent amb el comprador. Aquest contracte mantindria l'escriptura en custòdia fins que els fons del comprador es presentin correctament. El venedor estalvia diners, ja que no ha de pagar a un intermediari, i el comprador obté la casa molt abans del que l'hauria fet d'una altra manera.

- **Cadena de subministrament:** Podria dir-se que una de les implementacions més populars de la tecnologia blockchain i els contractes intel·ligents, en particular, és dins d'una cadena de subministrament.

  Botigues de comestibles, magatzems d'oficines, agricultors i més, tots tenen el seu lloc específic en una cadena de subministrament. Però amb la complexitat que estan agafant aquestes xarxes, a les empreses els resulta cada vegada més difícil fer un seguiment de la custòdia dels productes i seguir els pagaments, entre altres coses. Els contractes intel·ligents poden automatitzar i incentivar totes les parts de la cadena de subministrament per a augmentar la seva responsabilitat.

  Suposem que una empresa d'Europa vol comprar un enviament de mercaderies a un proveïdor d'Àsia.

  Podria automatitzar cada pas de la transacció, des de la comanda fins al lliurament, utilitzant un contracte intel·ligent. Tota la informació pertinent, com les especificacions del producte, la informació d'enviament, les condicions de pagament i els terminis de lliurament, s'inclourien en el contracte intel·ligent.

  Per a garantir que els articles s'ajusten a les expectatives del comprador, el contracte intel·ligent també inclouria condicions de qualitat i quantitat del producte. L'ús d'intermediaris, com a bancs o brokers, i les comissions associades a ells serien innecessaris perquè el contracte s’executa automàticament i no negociable.

  Els diners es guardaria en custòdia una vegada signat el contracte fins que el proveïdor certifiqui que s'han lliurat els productes. La cadena de blocs rastrejaria i guardaria els terminis de lliurament i la informació sobre els enviaments, la qual cosa donaria a totes dues parts una visibilitat i transparència totals.

5. IPFS

5.1 Què és?

IPFS o InterPlanetary File System, és un sistema d'arxiu descentralitzat que busca garantir la seguretat, privacitat i resistència a la censura de les dades. És un projecte amb un objectiu bastant clar: crear una xarxa de computadores d'abast global que permeti l'emmagatzematge d'informació de forma completament descentralitzada, amb una alta escalabilitat, i per descomptat, amb una gran resistència a la censura.

La idea és construir una xarxa P2P que permeti als qui formin part d'aquesta, emmagatzemar i distribuir informació de forma completament descentralitzada per tot el planeta. El sistema funciona sobre la base de la coneguda tecnologia de taula de hash distribuïda o DHT del qual IPFS presa algunes funcions per a la seva xarxa P2P.

La creació d'IPFS ve a resoldre una necessitat d'espai d'emmagatzematge de dades que està en constant expansió. Però el principal problema d'aquestes dades és que acaben en mans de tercers que generalment exploten els mateixos per a les seves diferents activitats econòmiques. Com per exemple, Google Drive és capaç d'analitzar el que escrius i guardar-ho en els seus servidors.

Per això neix IPFS, transformar la forma en com s'emmagatzemen les dades, deixant que aquests estiguin completament descentralitzats, i el control d'accés als mateixos sigui a les teves mans en tot moment. No sols això, IPFS permet que el nostre ordinador pot emmagatzemar dades d'una web i servir-los a qui els necessiti.

Això ajudaria a resoldre la necessitat d'espai d'emmagatzematge per a fer front a la demanda de tot el món. A més, ajudaria a descentralitzar la xarxa, i fins i tot, ens permetria guardar un historial complet d'aquella informació que ens interessi en una xarxa resistent i sense censura.

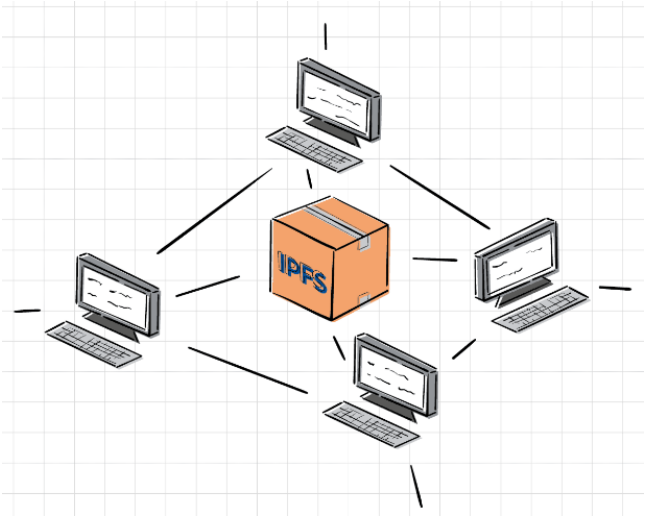


Figura 10: Imatge representativa de IPFS. [31]

5.2 Funcionament

Tots els recursos en IPFS estan identificats amb un identificador únic, anomenat CID. Aquests poden o bé començar per *Qm* o per *Bafy* com es pot observar en la següent figura. Són usats per adreçar el contingut i anomenar cada peça del IPFS.

Figura 11: Exemple de CID [32]

Es generen fent servir un hash del contingut i afegint prefixos que ens faciliten identificar la informació d'aquest. Un CID es compon de les següents parts:

- Primer prefix que ens indica la versió del mateix CID, sigui *Qm* o *Baffy.*
- Un mulitcodec que permet representar el tipo de codificació utilitzada per a serialitzar l'arxiu, això ens permet identificar el format de l'arxiu.
- Multibase, una codificació basi auto descriptiva seguida de les dades codificades.

Figura 12: Parts d'un identificador CID. [32]

Quan es puja un arxiu a IPFS, la xarxa el divideix en petites parts i les distribueix a través dels nodes. Aquestes parts es poden recuperar gràcies al hash associat a l'arxiu. Aquesta etiqueta permet a la xarxa identificar els nodes on s'emmagatzemen els paquets de dades. En altres paraules, per a recuperar un arxiu, n'hi ha prou amb introduir el seu hash en un navegador compatible, com per exemple Brave.

El procés és similar a buscar una pàgina web introduint el seu URL. Una vegada que s'identifica el hash, IPFS sol·licita als nodes la transmissió dels paquets a través de la connexió P2P. Un punt a destacar és que, igual que ocorre amb una xarxa blockchain, el contingut d'aquests arxius no es pot alterar en el procés. Si algun usuari intentés alterar un arxiu, canviaria el seu hash.

En realitat, es poden generar versions, connectant-les amb l'arxiu original (generant un historial del mateix totalment accessible). És una manera d'evitar els efectes secundaris derivats de la immutabilitat del contingut.

Figura 13: Divisió de l'arxiu en diverses parts. [32]

5.3 Avantatges i desavantatges

Entre els avantatges d'IPFS podem enumerar els següents:

- El sistema d'emmagatzematge està completament descentralitzat.
- La xarxa està construïda amb l'objectiu que sigui escalable.
- La xarxa pot resistir atacs, ja que es troba descentralitzada. D'aquesta forma es garanteix l'accés a la informació en cada moment.
- El seu ús és completament gratuït, i el codi font està disponible sota llicències de programari lliure.
- És extensible, el que permet que qualsevol persona pugui adaptar noves funcions sense problemes.

Per altra banda, si parlem sobre els desavantatges trobem:

- És un desenvolupament en plena evolució, i per això el seu ús en producció no és molt extens.
- És complex de fer servir per a usuaris inexperts en aquest tipus de sistema.
- No té extensions de privacitat per defecte.

6. Anàlisi de l'estat d'art de blockchain

6.1 La programabilitat de bitcoin

Bitcoin ha revolucionat el món financer al ser la primera criptomoneda descentralitzada. Al llarg dels anys, ha guanyat popularitat no sols com una forma de diners digitals, sinó també com una xarxa amb capacitats de programabilitat. Més enllà de les seves transaccions bàsiques, Bitcoin ofereix un llenguatge de programació anomenat Script, que permet la creació de condicions personalitzades per a desbloquejar i bloquejar fons. Aquesta funcionalitat ha obert noves possibilitats per al desenvolupament d'aplicacions descentralitzades i contractes intel·ligents en la xarxa.

Aquest apartat es centra en el llenguatge de programació Script i les aplicacions que sorgeixen d'ell. Examinant els fonaments i destacant l'estructura i característiques claus. A més de comparar la programabilitat limitada de Bitcoin amb llenguatges Turing complets com Solidity en Ethereum o Python.

Finalment, s'explorarà la possibilitat de crear smart contracts en Bitcoin i analitzant algun cas d'ús pràctics comparant la seva funcionalitat amb altres plataformes, abordant els seus límits i reptes d'escalabilitat.

6.1.1 Programabilitat limitada vs. Turing complet

Turing complet es refereix a la capacitat d'un sistema o llenguatge per a dur a terme qualsevol càlcul que pugui descriure's de manera algorítmica. Aquest terme prové del treball del matemàtic i científic de la computació britànic Alan Turing.

Un sistema o llenguatge de programació es considera Turing complet si pot simular una màquina de Turing, que és un model matemàtic d'un dispositiu de computació capaç de manipular símbols en una cinta infinita basant-se en un conjunt de regles. La complexitat de Turing es basa en la idea que qualsevol problema que pugui ser resolt algorítmicament també pot ser resolt per un sistema Turing complet.

En termes pràctics, un llenguatge o sistema Turing complet pot dur a terme càlculs complexos, implementar lògica condicional (com a declaracions if-*else), bucles (com a bucles for o while) i manejar la recursivitat (una funció que es diu a si mateixa). Pot representar i manipular dades, executar algorismes i resoldre problemes amb un conjunt d'instruccions prou poderós i expressiu.

Molts llenguatges de programació i sistemes computacionals, com Python, Java, C++ i JavaScript, es consideren Turing complets perquè tenen les característiques i capacitats necessàries per a simular una màquina de Turing i fer qualsevol tasca computable.

A continuació es realitzarà una comparació entre el llenguatge script de bitcoin i Python, per tal d'observar diferències entre codi i avantatges o desavantatges entre els dos.

	Bitcoin Script	**Python**
Turing	Incomplet	Complet
Sintaxi	Lineal i pila	Expressiva i llegible
Casos d'us	Transaccions	Web, dades, IA
Desenvolupament	Protocol Bitcoin (BIP)	Codi obert
Seguretat	Alta	Baixa
Execució	Comissions	Recursos (memòria, processament)

Taula 3: diferències entre Bitcoin Script i Python

Seguint l'exemple anterior en l'explicació del llenguatge Bitcoin Script sobre una transacció s'aprofitarà per a dur a terme una comparació entre els dos llenguatges per a observar les seves diferències.

OP_DUP OP_HASH160 <public_key> OP_EQUALVERIFY OP_CHECKSIG

6.1.2 Bitcoin Script: Fonaments i característiques

Quan parlem de Bitcoin Script, parlem d'un llenguatge de programació simple empleat en Bitcoin per al processament de les transaccions que es llegeix d'esquerra a dreta. Aquest està basat en una sèrie d'estructures lineals, conegudes com a pila (stack), que contenen dades existents en ordre LIFO (Last In – Firt Out). Cada instrucció en aquest llenguatge s'executa consecutivament una després de l'altra.

Aquest llenguatge no és Turing Complet pel fet que la seva funcionalitat és limitada i no pot realitzar bucles. En lloc de bucles, es basa en estructures de control condicionals (if-else), per a prendre decisions basades en l'estat actual de la pila de dades.

Pel que no és capaç de resoldre qualsevol classe de problema com les màquines Turing. No obstant això, aquesta limitació és intencional, ja que així s'evita l'entrada en un bucle infinit o sense fi i l'execució d'errors. On les parts malicioses del programa poden tenir la llibertat de crear operacions complicades per a consumir la taxa de hash i alentir el sistema de Bitcoin a través de bucles infinits.

En Bitcoin, per a donar comunicar les nostres idees són necessaris els codis d'operació (OP CODES), que serveixen per a diverses funcions. Com la manipulació de memòria, matemàtiques, anomenades a funcions, entre moltes altres.

Aquests OP_CODES són els que permeten efectuar les diferents operacions en Bitcoin i la seva programació de transaccions, com ho són el control de flux de dades, maneig de constants, maneig del stack o pila, maneig lògic, aritmètica, bloqueig de temps, pseudoparaules, operacions criptogràfiques, i paraules reservades. Pots veure una llista completa i actualitzada dels diferents OP_CODES directament en el codi de Bitcoin.

Figura 14: codis disponibles en bitcoin script

A part d'aquests codis, hi ha altres dos elements que són importants dins del Bitcoin Script: scriptSig i scriptPubKey. El primer, el scriptSig és el script de desbloqueig, que requereix d'una clau pública i una signatura digital. El segon, el scriptPubKey, és el script de bloqueig, que conté un hash de clau pública, també denominada direcció de Bitcoin.

6.1.3 Exemple d'un cas pràctic Bitcoin Script i Python

A continuació s'explicarà pas a pas un senzill exemple pràctic per veure cadascuna d'aquestes parts:

Un exemple de Bitcoin Script, seria la següent sentència, coneguda com scriptPubKey:

OP_DUP OP_HASH160 f6b137461f8bd00a52357670fd2b175b259937d7 OP_EQUALVERIFY OP_CHECKSIG

A més, a aquests codis els acompanya el següent scriptSig o script de signat.

3045022100ff0ce40d6cebf491ef78a5e9a36f4acfec0ba1b3814e6fbbbe184c7823e700470220245c951e7f9456aa9b375c0c795b2c81590d35e141136435ea0d8033657041e001

036e3e5a48123d88a9e325b7692594b18b0692d5ca7f6beff8fb8adaef54dd98fc

En primer lloc, observem el scriptSig. En aquests espais estan continguts dues dades importants, la clau pública de l'usuari i la seva signatura digital. Aquestes dades són els primers a introduir-se en la pila d'execució del Bitcoin Script. Quedant en primer lloc, la signatura i, en segon lloc, la clau pública.

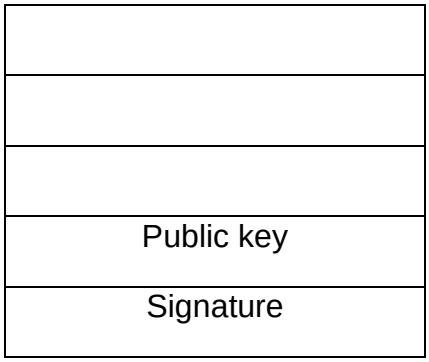


Figura 15: pila amb la clau publica i la signatura digital

Seguidament comença a executar-se, el script. La primera operació és el codi d'operació OP_DUP. Aquest codi ordena que es dupliqui el cim de pila d'instruccions. És a dir, la clau pública es duplicarà quedant el seu duplicat en tercer lloc. El segon codi d'operació és OP_HASH160. Aquest codi ordena que la clau pública duplicada sigui convertida usant les funcions SHA-256 i RIPEMD-160.

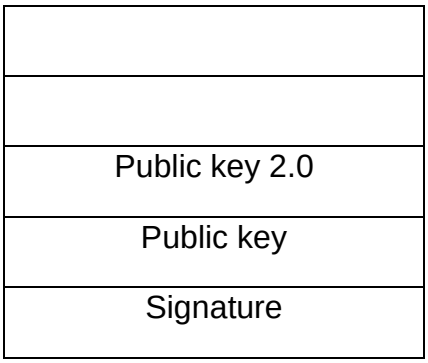


Figura 16: pila amb el duplicat de la clau pública

Això significa que es prendrà la clau pública i se li aplicarà un hash SHA-256. Al resultat d'aquesta operació li apliquem un hash RIPEMD-160. El seu resultat final és posat finalment en la pila. Seguidament, el script ens mostra un altre hash RIPEMD-160, que simplement és copiat en la pila per a donar pas a la següent operació.

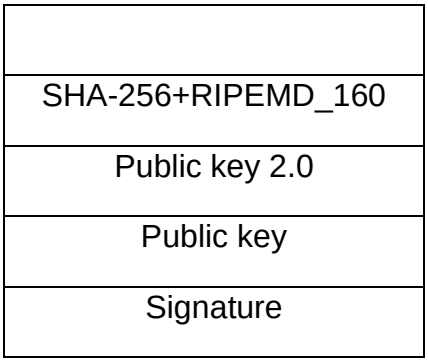


Figura 17: pila amb el resultat final dels hash

La següent operació en el script és OP_EQUALVERIFY. Aquesta prendrà els dos hashes RIPEMD-160 (el que ve de la clau pública del scriptSig i el que ve del Bitcoin Script) i verificarà que siguin iguals. Si l'operació és ”true”, seguirà l'execució del script, i en cas contrari, la transacció serà marcada com a invàlida.

Finalment, per a acabar l'operació, ens queda el codi OP_CHECKSIG, que ens indica que verifiquem la signatura digital del scriptSig. Si la signatura és igualment verificada de manera correcta, es dona com a completada tota l'operació i el seu resultat queda plasmat en la blockchain.

És a dir, cada vegada que realitzem una transacció en Bitcoin, els nostres moneders escriuen una programació semblant a aquesta i els scripts són interpretats pels nodes. Això succeeix quan els nodes reben un bloc vàlid per part dels miners. De fet, aquest petit exemple és d'una transacció real que pots verificar aquí.

La principal raó pel que aquest sistema té un llenguatge d'scripting és per a una major flexibilitat. Gràcies a Bitcoin Script és possible crear una infinitat d'aplicatius avançats. Per exemple, crear moneders i direccions multifirma. A més, parlem d'un sistema que és més fàcil de mantenir, ja que la compatibilitat de Bitcoin mai ha estat trencada i versions antigues de moneders poden seguir sent utilitzades sense problemes, perquè poden programar transaccions usant Bitcoin Script.

Degut al seu disseny minimalista i enfocat a la seguretat té certes limitacions com: l'absència de Turing i la falta de suport per a operacions fora de cadena, però solucions com Lighting Network permeten aquestes operacions fent servir contractes intel·ligents de Bitcoin.

Seguidament es mostrarà com es pot implementar el mateix exemple anterior però amb Python, un llenguatge de Touring complet.

```python
import hashlib
import ecdsa

def hash160(public_key):
    sha = hashlib.sha256(pubkey).digest()
    ripe = hashlib.new('ripemd160', sha).digest()
    return ripe

def verify_transaction(tx_output, public_key):
    hash160_pubkey = hash160(pubkey)
    return tx_output == hash160_pubkey

# Dades de la transacció i del destinatari
tx_output = "xxxxxxxxxxxxxxxxxxxxxxxx"
public_key = "yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy"

if verify_transaction(tx_output, public_key):
    print("El destinatari pot reclamar")
else:
    print("El destinatari no pot reclamar.")
```

Figura 18: codi en Python sobre transacció de bitcoin

En aquesta implementació es necessitar importar els mòduls necessaris per a realitzar funcions hash i manejar operacions criptogràfiques, a més de proporcionar dades com la clau publica i la transacció.

Definim una funció *hash160* que pren com a entrada la clau publica. En primer lloc realitza un hash_256 sobre les dades i després calcula un hash ripemds-160 sobre el resultat anterior, aquesta part seria el codi OP_HASH160.. Després, s'observa una altra funció que verifica la transacció utilitzant la clau publica. Aquesta funció s'usa per a realitzar OP_CHECKSIG.

Si totes les verificacions son exitoses imprimim que el destinatari pot reclamar, en cas contrari, s'imprimirà missatges d'error.

En aquest exemple no es mostra com s'implementa un sistema de transaccions complet, però sí que es pot demostrar com amb una línia de codi amb Bitcoin Script fent servir els OP_CODEs, podem aconseguir el desitjat sense implementar tantes línies com en Python. Cal remarcar, però que mentre Python és un llenguatge de propòsit general i no està directament relacionat amb la xarxa Bitcoin.

Per a donar com a finalitzat aquest apartat es comentarà una comprovació del que es pot fer en Python i no en Bitcoin Script, en aquest cas es tractarà el Rolling hash. És una tècnica per a calcular un valor hash de manera eficient a mesura que es desplaça o es recórrer una seqüència de dades.

Aquesta tècnica és especialment útil en aplicacions que requereixen una comparació ràpida i eficient de subcadenes, com la de duplicació de dades, la cerca de patrons i la detecció de col·lisions.

La idea darrera un Rolling hash és dividir les dades en finestres i calcular el valor del hash de cada finestra en funció del valor del hash de la finestra anetiror del nou element afegit a la finestra actual. Cosa que permet no haver de fer el càlcul complet des de zero cada vegada que la finestra es desplaci.

L'algorisme més utilitzat per a Rolling hashs és l'algorisme Rabin-Karp, que utilitza una funció de hash polinòmica per calcular el valor de hash per a la finestra de dades. La funció hash polinòmica utilitza els coeficients d'un polinomi per representar la finestra de dades, i el valor hash es calcula mitjançant l'avaluació del polinomi en un punt fix en un cos finit. El valor del hash es pot actualitzar eficientment eliminant el primer coeficient del polinomi i afegint el següent coeficient de la finestra mòbil, que es pot fer en temps constant.

A continuació es detalla com funciona la tècnica implementada en Python:

```python
def rolling_hash(s: str, window_size: int, base: int = 26, mod: int = 10**9 + 7):
    """

    :param s: cadena d'entrada
    :param window_size: tamany de la finestra
    :param base: la base per a la funció polinomial
    :param mod: mòdul per a la funció polinomial
    :return:     una llista dels valor hashs de totes les subcadenes de llargada de windows_size en s

    """
    n = len(s)
    power = [1] * (n + 1)
    hash_values = [0] * (n - window_size + 1)

    # calcular previament les potencia de la base mòdul del mod
    for i in range(1, n + 1):
        power[i] = (power[i - 1] * base) % mod

    # calcula el valor hash de la primera finestra
    current_hash = 0
    for i in range(window_size):
        current_hash = (current_hash * base + ord(s[i])) % mod

    hash_values[0] = current_hash

    # calcula els valors hash de la resta de subcadenes
    for i in range(1, n - window_size + 1):

        # elimina la contribució del primer caràcter a la finestra actual
        current_hash = (
            current_hash - power[window_size - 1] * ord(s[i - 1])) % mod

        # desplaça la finestra un caràcter i afegeix el caràcter nou al hash
        current_hash = (current_hash * base + ord(s[i + window_size - 1])) % mod

        hash_values[i] = current_hash

    return hash_values
```

Figura 19: codi de la funció Rolling hash en python

Aquesta funció inclou una cadena s, una mida de finestra window_size i dos arguments opcionals base i mod. L'argument base s'utilitza com a base per a la funció hash polinomial, i el mod és el mòdul utilitzat per reduir els valors hash.

La funció inicialitza primer algunes variables, inclosa una potència de llista que emmagatzema les potències de la base mòdul del mod, una llista hash_values que emmagatzema els valors hash de totes les subcadenes de longitud window_size en s, i una variable current_hash que emmagatzema el valor hash de la finestra actual.

Aleshores, la funció utilitza un bucle for per calcular prèviament les potències de la base mòdul del mod. A continuació, calcula el valor hash de la primera finestra fent servir un altre bucle for que itera sobre els caràcters de la primera finestra.

Finalment, la funció utilitza un altre bucle for per calcular els valors hash de la resta de subcadenes. Primer elimina la contribució del primer caràcter a la finestra actual, desplaça la finestra un caràcter i afegeix el caràcter nou al hash. A continuació, emmagatzema el valor hash de la finestra actual a la llista hash_values.

La funció retorna la llista hash_values , que conté els valors hash de totes les subcadenes de longitud window_size en s.

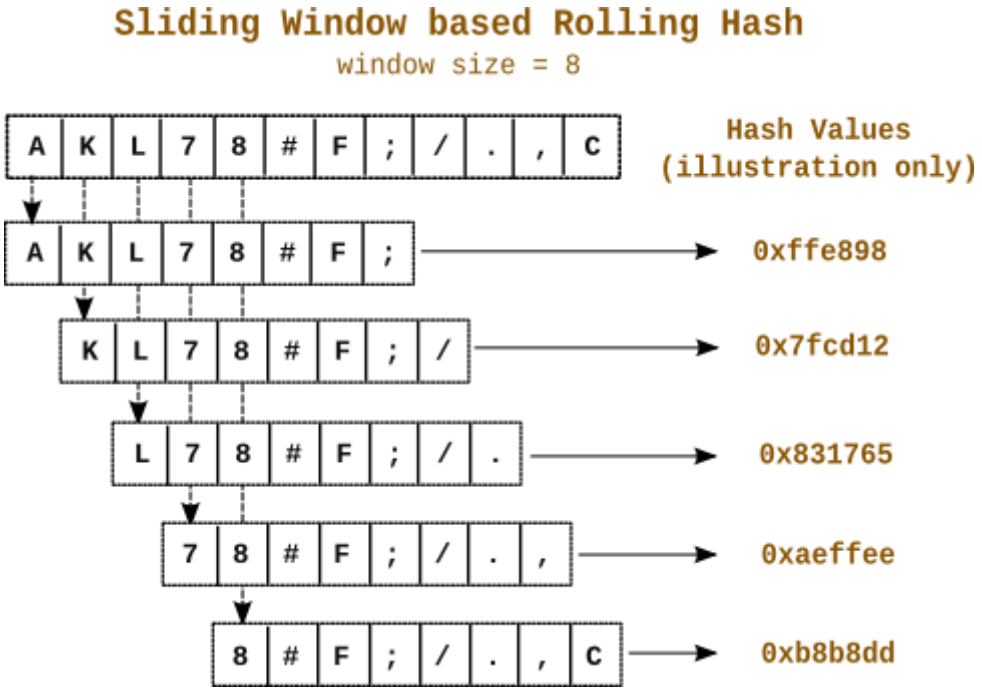


Figura 20: funcionament gràfic del Rolling hash [57]

Bitcoin Script és un llenguatge de programació dissenyat específicament per a operacions de script en transaccions Bitcoin. No obstant això, ofereix una sèrie de limitacions i restriccions que fan que sigui difícil o pràcticament impossible implementar la funcionalitat de "rolling hash".

Bitcoin Script es va dissenyar com un llenguatge de script molt senzill amb un conjunt limitat d'accions i funcions. Aquests passos se centren principalment a verificar les condicions per al desbloqueig de la sortida de la transacció, com ara la verificació de signatures criptogràfiques. No hi ha passos per realitzar càlculs hash complexos o manipular dades en temps real.

No té les variables avançades i les estructures de dades que es poden requerir per implementar la funció. Cosa que fa que sigui difícil mantenir una finestra desplaçant de dades i calcular de manera iterativa el hash a mesura que canvien les dades.

A mésBitcoin té un fort enfocament en la seguretat i la confiança. La implementació d'operacions més complexes o l'ús de funcionalitats no estàndard poden comprometre la seguretat i la confiança de la xarxa perquè augmenten la superfície d'atac i la complexitat del sistema.

En resum, les limitacions i restriccions a Bitcoin Script fan que sigui molt difícil o impossible implementar la funcionalitat "rolling hash" directament en aquest llenguatge. La seguretat i la eficiència són dos punts molt importants a destacar, per tant s'ha de limitar a les estrictes necessitats de les transaccions i operacions relacionades amb la cadena de blocs.

6.1.4 Programabilitat en els smart contracts de Bitcoin

Un smart contract és capaç d'executar-se i fer-se complir per si mateix, de manera autònoma i automàtica, sense intermediaris ni mediadors. Eviten la interpretació perquè no és verbal o escrit en els llenguatges que parlem. Els smart contracts es tracten de scripts escrits amb llenguatges de programació. Això vol dir que els termes del contracte són pures sentències i comandos en el codi que el forma.

Bitcoin té smarts contracts i sempre els ha tingut. Més específicament, aquests smart contracts en Bitcoin reben el nom de Bitcoin Script i són els responsables que puguem mobilitzar valor dins de la xarxa BTC. I estan programats amb Bitcoin Script, explicat en el punt 6.1.1, el problema és que es tracta d'un llenguatge Turing incomplet, per això es diu que Bitcoin no té smart contracts, però el fet que això sigui així és perquè es tracta d'un llenguatge molt segur a diferència de Solidity (smart contracts amb Ethereum).

L'anterior es torna molt més destacat quan compares un codi en Bitcoin Script al costat d'un codi en Solidity, senzillament Solidity és humanament llegible i fàcil de treballar. Tanmateix, això és una cosa que els desenvolupadors de Bitcoin coneixen i han millorat al llarg dels anys, i entre aquestes millores el protocol RGB presenta característiques que són úniques.

El protocol RGB ofereix nombroses possibilitats per a Bitcoin. No obstant això, l'avantatge més destacat és que funciona amb base en un esquema de

validació del costat client. Això significa que el protocol està ideat perquè cada usuari pugui veure, executar i verificar un contracte intel·ligent. Tot això sense la necessitat de processar-ho en la mateixa xarxa.

Gràcies a aquest sistema, l'execució de cada contracte es produeix únicament entre els usuaris interessats en aquest, sense sobrecarregar la xarxa. A més, RGB permet que els smart contracts s'executin fos de la cadena de blocs, és a dir, off-chain. D'aquesta manera s'evita el creixement excessiu de la blockchain en si.

La idea principal després de la implementació del protocol és que els contractes intel·ligents no s'emmagatzemin en la cadena de blocs i així evitar sobrecarregar la xarxa. Només necessiten accés els usuaris que interactuen amb el contracte intel·ligent. En fer-ho, poden usar els seus recursos informàtics per a executar-ho, i quan acaben d'executar, emmagatzemen la sortida del contracte i l'estat del mateix en la cadena de blocs.

Aquest model s'encarrega d'eliminar les complexes estructures com ho són les màquines virtuals distribuïdes. Un exemple d'aquestes estructures seria la Ethereum Virtual Machine. D'aquesta manera, els contractes són molt més flexibles i el cost per interacció es torna molt més baix, eliminant problemes com el gas. Això facilitaria el desenvolupament de DApps i la resolució de problemes. A la mateixa vegada es mantindria una major seguretat en el sistema i seria més difícil manipular-lo.

Una altra dels avantatges d'aquesta implementació, que de fet és de les més importants, és que tot el sistema es construeix sobre els fonaments actuals de Bitcoin i Lightning Network. La implementació de RGB fa possible agregar contractes intel·ligents a Bitcoin, sense interferir en el funcionament de Bitcoin.

En mode RGB, no s'utilitza la programació imperativa. Per tant, un esquema és una definició declarativa d'un contracte intel·ligent, no un requisit obligatori. Per tant, el canvi de paradigma de la programació Ethereum a la programació RGB és el mateix canvi de paradigma de la programació imperativa convencional a la programació funcional o declarativa.

Els contractes intel·ligents RGB són canviants d'estat (DAG) on només es coneix una part del gràfic. Un esquema és essencialment un conjunt de regles per a l'evolució dels gràfics, però en lloc de manipular tot el gràfic, s'aplica als nodes individuals del gràfic i descriu de manera declarativa com pot canviar amb el temps. Cada node té les seves pròpies regles i el gràfic resultant és una aplicació d'aquestes regles.

La xarxa Bitcoin dona suport a una àmplia varietat de contractes intel·ligents. Aquest inclouen:

- **Pay-to-Public-Key-Hash (P2PKH):** L'escriptura P2PKH és un dels contractes intel·ligents Bitcoin més senzills. Permet a algú enviar BTC a una adreça Bitcoin de manera que només el propietari de la clau privada corresponent al hash de la clau pública pot gastar el Bitcoin.

- **Scripts multi signatura:** Els scripts multi signatura són més complexos que els P2PKHs perquè requereixen múltiples signatures en lloc d'una. Això vol dir que el Bitcoin utilitzat per a una transacció només es pot gastar quan es proporciona el nombre requerit de signatures Per exemple, el 2-de-3 és una configuració multi signatura comuna que permet a tres parts amb tres claus públiques mantenir BTC com a grup. Requereix dues signatures perquè Bitcoin es gasti.

- **Operacions Bitcoin bloquejades temporalment:** Les transaccions Bitcoin bloquejades temporalment són vàlides després d'una durada especificada. A més, poden utilitzar-se per a modificar els requisits de despesa de la BTC. Per exemple, un guió pot exigir tres signatures per passar BTC abans d'un temps determinat. Després d'aquesta durada, només cal una signatura. Un tipus d'aquesta transacció també es fa servir a la Xarxa de Llamp de Bitcoin.

- **Pay-to-Script-Hash (P2SH):** L'estàndard P2SH ajuda els usuaris de Bitcoin a enviar BTC al hash de qualsevol script. Això redueix el cost d'enviar Bitcoin a un contracte intel·ligent complex mentre es manté la privacitat. Gràcies a l'actualització de SegWit, l'estàndard P2SH ara inclou Pay-to-Witness-Script-Hash (P2WSH).

- **Contractes intel·ligents Bitcoin a les plataformes de la capa 2:** Bitcoin també pot executar contractes intel·ligents amb protocols com ara Lightning Network (LN), que es construeix a sobre de la xarxa Bitcoin. LN es basa en transaccions de signatura múltiple anomenades contractes híbrids bloquejats en temps (HTLC) per permetre micropagaments de bitcoin barats i instantanis. HTLC assegura que aquells que ajuden als pagaments directes entre diferents parts paguen una petita tarifa, però no tenen accés a cap bitcoin. Altres capes de Bitcoin, com Liquid Network o RSK, depenen de les funcions de contracte intel·ligent de Bitcoin per proporcionar casos d'ús addicionals.

6.1.5 Escalabilitat i limitacions en la programabilitat de Bitcoin

La preocupació de la capacitat d'una xarxa per gestionar un augment del volum de transaccions sense reduir la seva velocitat o eficiència es coneix com a escalabilitat.

L'escalabilitat limitada de Bitcoin afecta directament la seva programabilitat. Els contractes intel·ligents i les aplicacions descentralitzades que requereixen un gran nombre de transaccions per segon poden ser difícils d'executar de manera eficient a la xarxa Bitcoin.

Per una banda, el protocol Bitcoin limita la mida del bloc a 1 MB, la qual cosa significa que només es pot processar un nombre limitat de transaccions en cada bloc. Com a resultat, la capacitat de la xarxa per processar transaccions és limitada i pot provocar congestió i retards en èpoques de gran demanda.

Per altra banda, el procés de verificació de les transaccions de Bitcoin pot ser lent i costós a causa de la competència entre els miners i del seu algorisme de consens PoW, que requereix una gran quantitat de poder computacional i, per tant, energia. A mesura que més usuaris s'uneixen a la xarxa i envien més transaccions, la quantitat de mineria de dades que s'han de processar augmenta significativament, cosa que pot provocar retards i reduir la velocitat i l'eficiència de les transaccions, com també ho fa el consum energètic, plantejant preocupacions ambientals i de sostenibilitat a llarg termini.

Els casos d'ús que depenen d'una execució ràpida i eficient de contractes intel·ligents, com ara sistemes de votació o jocs en línia, poden ser difícils a causa dels temps de confirmació més llargs i de les transaccions limitades amb la capacitat de processament. A més, l'escalabilitat limitada podria afectar l'adopció i l'interès dels desenvolupadors per crear aplicacions a la xarxa Bitcoin. La comunitat de desenvolupament ha d'abordar els problemes d'escalabilitat per crear un entorn més propici per al desenvolupament avançat de Bitcoin.

En resum, les limitacions actuals de Bitcoin tenen un impacte significatiu en les possibilitats de desenvolupament en la xarxa. A mesura que s'exploren aplicacions més complexes i es busquen solucions innovadores, és important comprendre com aquestes limitacions influeixen en el panorama general de desenvolupament. Aquestes són:

- **Llenguatge de programació limitat:** com s'ha explicat anteriorment Script de Bitcoin és simple comparat amb llenguatges Turing complets com Solidity. Això limita la complexitat de contractes intel·ligents i aplicacions.

- **Temps de confirmació i tarifes variables:** a causa de l'escalabilitat limitada, els temps de confirmació de les transaccions pot ser variable i les tarifes de transaccions poden augmentar.

- **Falta d'estàndards per a contractes intel·ligents:** A diferència d'Ethereum, que ha establert estàndards com ERC-20 i ERC-721 per a fitxes i actius digitals, Bitcoin no té estàndards similars per a aplicacions i contractes intel·ligents. Això pot dificultar la interoperabilitat entre diferents contractes i aplicacions descentralitzades a la xarxa Bitcoin.

Figura 21: Gràfic comparatiu entre Bitcoin i altres sistemes de pagament [43]

Aquestes limitacions i reptes comentats en aquest punt, porten a parlar de les solucions de Bitcoin Layer-2. Protocols creats sobre la capa base de Bitcoin, desenvolupats per a proporcionar escalabilitat a la xarxa mitjançant el processament de transaccions fora de la cadena principal de blocs. A més, de poder proporcionar funcionalitats addicionals a Bitcoin.

Els avantatges d'aquestes solucions són:

- **Escalabilitat millorada:** millorar a superar les limitacions de Bitcoin
- **Programabilitat millorada:** a través de la implementació de la funcionalitat d'smart contracts.
- **Seguretat:** ja que la capa base es manté intacta.
- **Capacitats ampliades:** noves utilitats com DeFi, NFT (Non Fungible Token).

Començant per la primera solució, **Lightning Network** és un protocol de pagament dissenyat com una solució d'escalabilitat per reduir la congestió de la xarxa quan es processen micropagaments de Bitcoin fora de la cadena mitjançant canals de pagament. La xarxa Lightning permet crear canals directament entre dues parts que volen fer transaccions, evitant potencialment la cadena de blocs principal.

La xarxa Lightning permet realitzar múltiples transaccions sense necessitat de consens i aprovació global a la cadena principal de Bitcoin. Com a resultat, les transaccions són gairebé instantànies i les tarifes són extremadament baixes. Fins ara, ha proporcionat una solució eficaç al problema d'escalabilitat de Bitcoin en poder gestionar fàcilment un gran nombre de transaccions. Les transaccions de diverses parts i l'encaminament de pagaments es produeixen de manera segura a la xarxa Lightning mitjançant contractes híbrids de bloqueig de temps (HTLC), que són contractes intel·ligents que permeten als destinataris rebre fons després que es compleixin determinades condicions en un moment específic (o bloqueig).

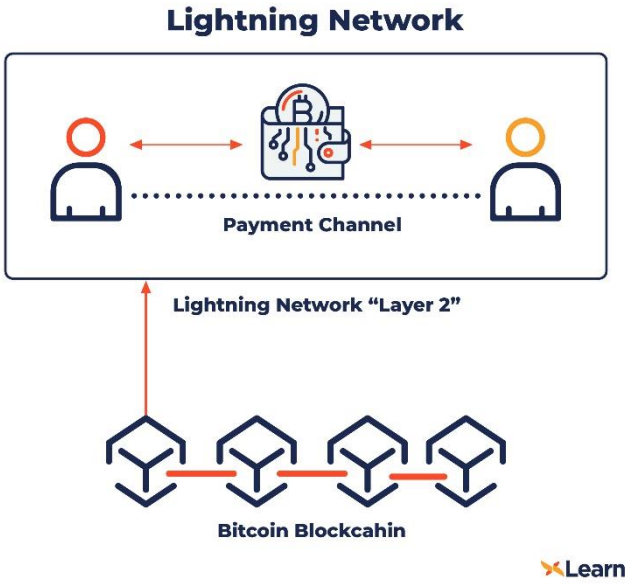


Figura 22: funcionament Lighting Network [44]

Una altra solució semblant a la primera és **Rootstock**, permet transaccions més ràpides i econòmiques processant-les fora de la cadena de blocs principal de Bitcoin. És una plataforma de contracte intel·ligent basada en el model Ethereum Virtual Machine (EVM), però connectada a la cadena de blocs de Bitcoin mitjançant una cadena lateral.

La cadena de blocs Rootstock utilitza un mecanisme de consens de prova de treball (PoW) per extreure blocs mitjançant la mineria de fusió. En aquest procés, els miners poden fer servir el mateix algorisme hash per extraure Bitcoin i Rootstock alhora i compartir la mateixa potència de càlcul. El temps de confirmació del bloc a Rootstock és d'uns 30 segons i el rendiment és d'uns 10-20 tps (transaccions per segons).

Com a plataforma de contracte intel·ligent, Rootstock amplia la funcionalitat de Bitcoin introduint la funcionalitat de contracte intel·ligent per a una major programabilitat i usabilitat. També facilita la interoperabilitat amb Ethereum mitjançant la màquina virtual RSK (RVM) basada en EVM. Admet l'execució de contractes intel·ligents d'Ethereum en Rootstock i la interoperabilitat amb les aplicacions d'Ethereum. RVM també permet als desenvolupadors codificar en el llenguatge de programació de contracte intel·ligent d'Ethereum, Solidity.

Una altra característica clau de Rootstock és l'ús del RSK Infrastructure Framework (RIF), que proporciona un conjunt de serveis d'infraestructura descentralitzada que els desenvolupadors poden utilitzar per crear i desplegar aplicacions descentralitzades (dapps) a la xarxa Rootstock.

I per últim, **Liquid Network** és una cadena lateral dissenyada per millorar el rendiment i la funcionalitat de la xarxa Bitcoin. Liquid va ser pensat inicialment com una solució escalable i una plataforma d'emissió d'actius que funciona independentment de Bitcoin, amb el seu propi registre global i mecanisme de consens.

Liquid és capaç de confirmar transaccions en menys de dos minuts, té un temps de bloc de 60 segons i té un objectiu de dos blocs. Això fa que el rendiment de les seves transaccions sigui significativament més ràpid que la xarxa Bitcoin, la qual cosa és una consideració clau per als usuaris que tinguin intenció de fer transaccions sensibles al temps.

Liquid funciona mitjançant un mecanisme de consens anomenat "federació forta" que es basa en un subconjunt dels seus membres (un grup d'organitzacions locals de criptomoneda) (anomenats "oficials") per aprovar transaccions i signar blocs. Aquest sistema redueix el temps d'aprovació de Liquid, mentre que l'estructura centralitzada s'encarrega de la gestió de la cadena i millora el rendiment.

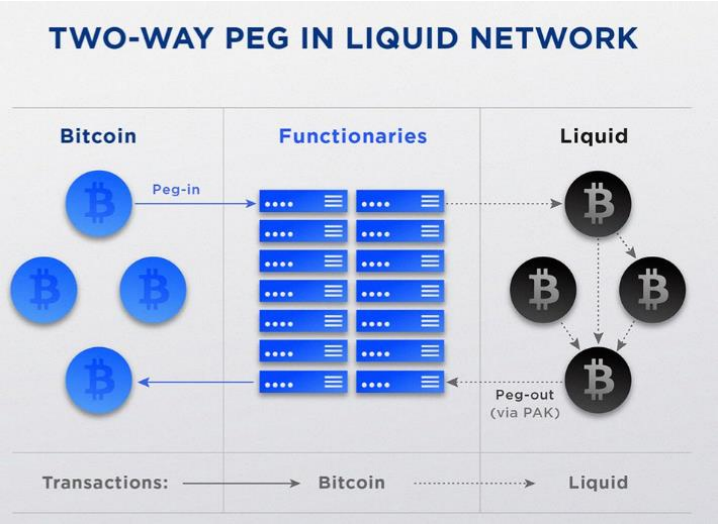


Figura 23: Funcionament Liquid en una transacció [42]

	Lightning Network	**Rootstock**	**Liquid**
Token natiu	No	RBTC	L-BTC
Smart contracts	No	Si	Si
Llenguatge de programació	Múltiples	Solidity	Simplicity
Algorisme de consens	No	PoW	Strong Federations
TPS	Milions	10-20	7-10
Objectiu	Quasi instantani	12 blocs de confirmació (6 mins)	2 blocs (6 mins)
Compatible EVM	No	Si	No

Taula 4: Comparació entre les tres solucions a l'escalabilitat de Bitcoin

6.2 La programabilitat d'Ethereum

A diferència del seu predecessor, Bitcoin, Ethereum no es va limitar a ser una simple moneda digital, sinó que es va proposar ser una plataforma completa que permetés la programació i execució de contractes intel·ligents i aplicacions descentralitzades (dApps). Aquesta introducció explora els antecedents que van portar a la creació de Ethereum, ressalta la importància de la programabilitat en l'ecosistema de criptomonedes.

Bitcoin va revolucionar el sistema financer en permetre les transferències de valor peer-to-peer, i Ethereum fa aquesta revolució un pas més enllà introduint la programabilitat a la cadena de blocs. La programabilitat significa la capacitat d'executar codi personalitzat a la cadena de blocs, la qual cosa obre un món de possibilitats per a aplicacions descentralitzades, contractes intel·ligents i sistemes autònoms.

La programabilitat permet automatitzar processos, crear fitxes personalitzades i implementar sistemes complexos sense intermediaris. Aquesta capacitat ha revolucionat les indústries des de les finances i els jocs fins a la identitat digital. La programabilitat d'Ethereum ha estat una revolució per a la indústria financera descentralitzada (DeFi) i ha impulsat la innovació en el desenvolupament d'aplicacions blockchain.

6.2.1 Llenguatge de programació Solidity

Solidity és un llenguatge de programació d'alt nivell orientat a objectes per escriure contractes intel·ligents a la plataforma Ethereum. És un dels llenguatges de programació més populars per escriure contractes intel·ligents per la seva senzilla sintaxi. Aquest llenguatge està inspirat en altres llenguatges com C, Python o JavaScript.

Amb Solidity, els desenvolupadors poden escriure contractes intel·ligents per controlar els intercanvis de criptomoneda, la propietat dels actius digitals i altres tipus de transaccions. Els contractes intel·ligents escrits amb aquesta eina són immutables i s'executen automàticament, la qual cosa els converteix en una bona opció per a aplicacions descentralitzades que requereixen seguretat i transparència.

Solidity admet una varietat de tipus de dades, inclosos enters, booleans, cadenes, arranjaments i més. També permet definir estructures de dades personalitzades a través de structs i enums. Les estructures de control comuns, com a bucles i condicionals, són presents en Solidity per a permetre la implementació de lògica complexa en els contractes intel·ligents.

Les funcions en Solidity es defineixen amb la paraula clau "function" i poden prendre arguments i retornar valors. La gestió de la visibilitat i el control d'accés a les funcions és fonamental en Solidity, ja que determina qui pot interactuar amb el contracte i de quina manera.

Com funciona Solidity? L'execució de contractes intel·ligents en Ethereum és implementada per una màquina virtual (EVM), encarregada d'executar els nodes de la xarxa i responsable de verificar els contractes. Aquests estan dissenyats per ser segurs i transparents, un cop desplegats a la blockchain d'Ethereum son immutables i no poden ser modificats.

Com que la sintaxi és semblant a altres llenguatges, inclou característiques de programació orientada a objectes com herència, encapsulament i polimorfisme, el que permet a aquest llenguatge ser complet i tenir la capacitat per a desenvolupar tot tipus d'aplicacions.

6.2.2 Exemple d'un cas pràctic Solidity i Python

A continuació, observarem un exemple pràctic de contracte intel·ligent escrit amb Solidity explicat, i comparat, amb un llenguatge Turing complet com és Python.

```
1  pragma solidity ^0.8.0;
2
3  contract TransaccionEther {
4      function transferir(address payable destino) public payable {
5          require(msg.value > 0, "Debe enviar ether");
6          destino.transfer(msg.value);
7      }
8  }
9
```

Figura 24: contracte intel·ligent amb Solidity

Aquest contracte compta amb una funció anomenada transferir que pren com a paràmetre l'adreça a la qual s'enviarà el ether i ha de ser una adreça vàlida i "pagable". La funció també es declara com payable per a permetre rebre ether. Quan algú crida a la funció transferir i envia ether juntament amb la transacció, es verifica que la quantitat enviada sigui major a zero. Després, el ether es transfereix a la direcció especificada.

Un cop escrit el contracte es compilaria amb Remix per exemple, i generarà un bytecode i el ABI (Application Binary Interface) necessaris per a desplegar i interactuar amb el contracte. A continuació, s'ha de desplegar el contracte fent servir Remix o altres plataformes també com Truffle, Hardhat... un cop s'hagi desplegat ja es pot interactuar amb el contracte fent servir el moneder Ethereum o una aplicació de línia de comandes com "geth" o llibreries com web3.

En Python aquest mateix exemple seria:

```python
from web3 import Web3

w3 = Web3(Web3.HTTPProvider('http://localhost:8545'))

compte_origen = '0x... '
compte_desti = '0x... '

if w3.isConnected():
    # Crear una transacció
    transaccion = {
        'to': compte_desti,
        'value': w3.toWei(0.1, 'ether'),  # 0.1 ether en wei
        'gas': 21000,
        'gasPrice': w3.toWei('50', 'gwei'),  # Precio del gas en gwei
        'nonce': w3.eth.getTransactionCount(compte_origen),
    }

    # Firmar y enviar la transacció
    clau_privada = '0x... '
    transaccion_firmada = w3.eth.account.signTransaction(transaccion, clau_privada)
    tx_hash = w3.eth.sendRawTransaction(transaccion_firmada.rawTransaction)

    print(f"Transacció enviada. Hash de transacción: {tx_hash.hex()}")
else:
    print("No es pot connectar a la xarxa Ethereum.")
```

Figura 25: codi per realitzar una transacció d'ether en Python

Comparant els dos codis, s'arriba a la conclusió que la principal diferència està en la interacció i automatització. En Python estàs realitzant una transacció directa entre dos comptes fent servir una llibreria externa, mentre que en Solidity, s'utilitza un contracte programat per a automatitzar aquesta lògica de la transacció en la cadena de blocs. A més, destacar la diferència de numero de línies entre un codi i l'altre, amb Solidity és més simple.

6.2.3 Programabilitat en els smart contracts d'Ethereum

La programabilitat és un dels aspectes més destacats i poderosos dels contractes intel·ligents en la xarxa Ethereum.. Aquesta programabilitat permet una àmplia gamma de casos d'ús i aplicacions en diverses indústries. A continuació una anàlisi exhaustiva de la programabilitat en els smart contracts d'Ethereum:

- **Flexibilitat en la lògica de les immobiliàries:** L'habitual en aquest àmbit és que la compra d'una casa sigui un procés lent i burocràtic. Això pot canviar amb l'ús de la blockchain, perquè en els contractes intel·ligents s'acorden, inclouen i resguarden les clàusules pactades prèviament entre les parts involucrades, i les mateixes s'executen de manera automàtica, sense la intervenció de tercers.

  La cadena de blocs permet la liquidació en temps real de les transaccions, reduint el risc, però també limitant la capacitat de cancel·lar transaccions. S'elimina l'entorn sense confiança, és a dir, com que la cadena de blocs es basa en proves criptogràfiques, permet a les dues parts realitzar aquestes transaccions directament entre ells sense la necessitat de tenir un tercer pel mig.

  Encara no s'ha generalitzat l'ús de Smart Contracts en el sector immobiliari. Encara hi ha buits legals en la majoria dels països. Però, segurament, no faltarà molt temps perquè siguin part del dia a dia en el món immobiliari.

- **Tokens i estàndards de tokens:** Un token és un actiu digital que pot representar qualsevol cosa, des de valors i altres productes financers, col·leccionables virtuals o actius físics del món real. Els tokens són creats mitjançant un Smart contract en una plataforma com Ethereum, la qual a més permet transferir aquests tokens de manera segura a través d'internet. Hi ha dos tipus de tokens: els fungibles i no fungibles.

  Els tokens fungibles són actius digitals que poden ser reemplaçats per un altre actiu idèntic. En canvi, els tokens no fungibles (Non-Fungible Token) són actius digitals que no poden ser reemplaçats per un altre actiu idèntic. Cada NFT és únic i, per tant, tenen un valor més alt que els tokens fungibles. La fungibilitat és la capacitat d'un actiu de ser intercanviable amb un altre actiu idèntic. Un bé o servei és fungible si pot ser reemplaçat per un altre bé o servei idèntic. Per exemple, un barril de petroli és intercanviable amb qualsevol altre barril de petroli.

No fungible significa que cada token criptogràfic és únic i no intercanviable entre si. Els tokens no fungibles (NFT) són actius digitals que no poden ser reemplaçats per un altre actiu idèntic. Cada NFT és únic i, per tant, tenen un valor més alt que els tokens fungibles, per exemple:

- Actius físics (cases, cotxes, etc)
- Actius virtuals (col·leccionables, art)
- Actius de “valor negatiu” (préstecs, cargues)

Per últim, es destacarà un concepte molt important en Ethereum que són els ERCs, Ethereum Request for Comments. Són propostes generades per la comunitat de desenvolupadors de Ethereum amb l’objectiu d’impulsar aquesta plataforma generant interoperabilitat i facilitant a les aplicacions que fan us de tokens. Es tractaran els dos principals estàndards:

- **ERC-20:** encara que la seva funcionalitat sigui bàsica, és l'estàndard més utilitzat i amb major rellevància a causa de la seva gran interoperabilitat en l'entorn.

  Proporciona els mètodes de transferència de tokens (transfer), aprovació o autorització d'ús dels teus tokens a una altra direcció d’Ethereum (aprove), transferència de tokens autoritzada des d'una altra direcció de Ethereum (transferFrom), consulta de balanç actual (balanceOf) i consulta de la quantitat de tokens possible a fer servir de part d'una altra direcció Ethereum (allowance).

  Per a caracteritzar el token s'ha d'assignar valor a una sèrie d'atributs en forma de variable. L'únic paràmetre obligatori és el número de total de tokens disponible (totalSupply), els altres com el nom (name), el símbol (sym) i el nombre de decimals (decimals) són opcionals, encara que importants si vols donar un major detall al teu token i distingir-lo dels altres. El nombre de decimals ofereix la capacitat subdividir una unitat de token en 18 decimals.

  Per a guardar els balanços s'usa una variable de tipus ‘mapping’, que relaciona una direcció d’Ethereum amb el seu corresponent balanç de tokens: (mapping (Address => uint256) balanços)

Amb tot això, un token ERC20 tindria la següent estructura:

```
abstract contract ERC20Snapshot is ERC20 {
  Counters.Counter private _currentSnapshotId;
  struct Snapshots { uint[] ids; uint[] values; }
  mapping (address => Snapshots) private _balancesSnapshots;
  Snapshots private _totalSupplySnapshots;

  function _snapshot() internal virtual returns (uint) {
    _currentSnapshotId.increment();
    uint currentId = _getCurrentSnapshotId();
    // ...emit Snapshot event...
    return currentId;
  }

  function balanceOfAt(address account, uint snapshotId) public view virtual returns (uint) {
    (bool snapshotted, uint value) = _valueAt(snapshotId, _balancesSnapshots[account]);
    return snapshotted ? value : balanceOf(account);
  }

  function totalSupplyAt(uint snapshotId) public view virtual returns (uint) {
    (bool snapshotted, uint value) = _valueAt(snapshotId, _totalSupplySnapshots);
    return snapshotted ? value : totalSupply();
  }

  function _beforeTokenTransfer(address from, address to, uint amount) internal virtual override {
    super._beforeTokenTransfer(from, to, amount);

    if (from == address(0)) {         // mint
      _updateAccountSnapshot(to);
      _updateTotalSupplySnapshot();
    } else if (to == address(0)) {    // burn
      _updateAccountSnapshot(from);
      _updateTotalSupplySnapshot();
    } else {                          // transfer
      _updateAccountSnapshot(from);
      _updateAccountSnapshot(to);
    }
  }

  // _valueAt, _updateAccountSnapshot, _updateTotalSupplySnapshot not shown
}
```

Figura 26: codi estàndard ERC-20 [49]

- **ERC-721:** és l'estàndard dels NFT. Una altra característica és la seva indivisibilitat, ja que es manté com a unitat singular, a diferència de l'ERC20. És, per tant, un token no consumible, però si intercanviable. Pot representar qualsevol cosa, des de la propietat d'una obra d'art fins a un préstec o una multa de trànsit.

  Cada token ERC721 està identificat per un ANEU únic no modificable (uint256, sencer de 256 bytes), i una direcció d'Ethereum. Implementa els mètodes, atributs i esdeveniments de l'ERC20 (name, symbol, totalSupply, balanceOf) amb l'objectiu de tenir compatibilitat entre aplicacions, encara que no faci el mateix ús d'ells.

  De moment, el seu ús és inferior a l'ERC20, encara que cal reconèixer que el seu futur resulta molt prometedor. En l'explorador 'Etherscan'

es pot veure que existeixen 262 tokens ERC721 desplegats en la xarxa principal d'Ethereum.

```
contract ERC721 is IERC721 {
    string private _name, _symbol;
    mapping(uint => address) private _owners; // Mapping from token ID to owner address
    mapping(address => uint) private _balances; // Mapping owner address to token count
    mapping(uint => address) private _tokenApprovals; // Mapping from token ID to approved address
    mapping(address => mapping(address => bool)) private _operatorApprovals; // Mapping from owner to
        approved operators

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

    function balanceOf(address owner) public view virtual override returns (uint) { return _balances[
        owner]; }
    function ownerOf(uint tokenId) public view virtual override returns (address) { return _owners[
        tokenId]; }
    function name() public view virtual override returns (string) { return _name; }
    function symbol() public view virtual override returns (string) { return _symbol; }

    function approve(address to, uint tokenId) public virtual override {
        address owner = ownerOf(tokenId); require(to != owner, "approval to current owner");
        require(_msgSender() == owner or isApprovedForAll(owner, _msgSender()), "caller is not owner
            nor approved");
        _approve(to, tokenId);
    }

    function getApproved(uint tokenId) public view virtual override returns (address) { return
        _tokenApprovals[tokenId]; }

    function isApprovedForAll(address owner, address operator) public view virtual override returns (
        bool) { return _operatorApprovals[owner][operator]; }

    function transferFrom(address from, address to, uint tokenId) public virtual override {
        require(_isApprovedOrOwner(_msgSender(), tokenId), "caller is not owner nor approved");
        _transfer(from, to, tokenId);
    }

    function _isApprovedOrOwner(address spender, uint tokenId) internal view virtual returns (bool) {
        address owner = ownerOf(tokenId);
        return spender==owner or isApprovedForAll(owner, spender) or getApproved(tokenId) == spender;
    }

    function _transfer(address from, address to, uint tokenId) internal virtual {
        require(ownerOf(tokenId) == from, "transfer from incorrect owner");
        require(to != address(0), "transfer to the zero address");
        _beforeTokenTransfer(from, to, tokenId);
        _approve(address(0), tokenId); // Clear approvals from the previous owner
        _balances[from] -= 1; _balances[to] += 1; _owners[tokenId] = to;
        emit Transfer(from, to, tokenId);
    }

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

Figura 27: codi estàndard ERC-721 [50]

- **DeFi o finances descentralitzades:** busquen desenvolupar serveis financers tradicionals, però amb un major grau de transparència i descentralització. Aquests serveis tenen la finalitat de desenvolupar tot un ecosistema de petites eines que en conjunt formin una gran solució per a les finances que eliminin la necessitat d'institucions financeres centralitzades.

  Volen convertir les estructures de finances centralitzades que tenim ara, en estructures descentralitzades, sense tercers de confiança, executades sobre smart contracts o contractes intel·ligents, dins d'una blockchain on quedi reflectit de manera transparent un registre escrit i inalterable de cada acció realitzada.

  Totes les accions que es facin en les plataformes queden visibles i gravades de manera immutable també en la blockchain. Potser un no té el coneixement per a verificar la fiabilitat del servei DeFi, però a causa de la seva característica de transparència, milers de persones podran analitzar-lo i qüestionar-lo, alertant del seu mal funcionament si fos així.

  Alguns dels avantatges de les DeFi són: habilitar l'accés a serveis financers a milions de persones que no estan relacionades amb el banc. És una oportunitat per a desenvolupar i aconseguir llibertat financera. Permet que el finançament internacional d'empreses i projectes sigui més senzilla, les DeFi poden ajustar-se a necessitats d'un públic amb l'objectiu de portar inversions on les necessitin. Per últim, creen un punt de diversificació econòmica i desenvolupament.

  De què ens serveixen les DeFi en l'actualitat? Doncs els casos d'us més importants són:

    - **Sistemes de lending descentralitzats:** El sistema és senzill: si una persona desitja un préstec i vol usar de col·lateral o garantia les seves criptomonedes, pot fer-ho sense problemes. El sistema funciona de forma molt similar a les finances tradicionals. Però DeFi sol oferir millors condicions d'interès i generalment els préstecs solen aprovar-se gairebé de manera instantània. Enrere queden les hores en el banc, enviar documents digitals i esperar dies per a una resposta, amb DeFi basta interactuar amb la DApp, realitzar el dipòsit de garantia exigit i tindràs en el teu poder els diners que has requerit en préstec, i tot en uns minuts.

- **Mercats descentralitzats:** Creacions com els exchanges descentralitzats (DEX), els pools d'inversions, derivats financers, sistemes de stakings, mercats de predicció, i més són possibles gràcies al DeFi.

- **Sistemes de pagaments:** Les característiques d'aquestes plataformes els permeten ser un pont de confiança per a processar pagaments de diferents blockchain, fent ús d'una infraestructura externa, descentralitzada i autònoma.

- **Serveis bancaris i d'assegurança:** oferir serveis del tipus “bancari” sense ser exactament un banc. Per exemple, hi ha protocols DeFi que permeten als seus usuaris realitzar una determinada inversió. Però al cap d'un temps, pots rebre aquesta inversió amb un marge de guany, i tot gràcies als interessos que la mateixa ha generat. Per altra banda, sinó que també hi ha sistemes que permeten l'emissió de monedes estables (stablecoins), així com sistemes d'identificació digital i d'assegurances financeres.

Característiques	Tradicionals	Defi
Control del sistema	Govern i banc	Descentralitzat
Confiança	Banc i tercers	Sense intermediaris
Transferències	Fiat	Criptomonedes, tokens
Control	Bancs	Deute tokenitzat
Mercat	Exchanges tradicionals	Exchange descentralitzat

Taula 5: diferències entre finances tradicionals i DeFi

6.2.4 Escalabilitat i limitacions en la programabilitat d'Ethereum

L'escalabilitat i les limitacions són punts clau en el disseny i la implementació dels smart contracts en la xarxa Ethereum. A mesura que l'adopció d'Ethereum i les aplicacions descentralitzades (dApps) ha augmentat, s'han tornat evidents uns certs desafiaments que afecten la programabilitat i l'eficiència de la xarxa. Aquí s'examina detalladament l'escalabilitat i les limitacions en la programabilitat d'Ethereum.

A mesura que ha crescut el nombre de persones que utilitzen Ethereum, la cadena de blocs ha assolit algunes limitacions de capacitat. Això ha augmentat el cost de fer servir la xarxa, creant la necessitat de solucions. Hi ha diverses solucions investigades, provades i implementades que adopten diferents enfocaments per assolir objectius similars.

L'objectiu principal de l'escalabilitat és augmentar la velocitat de transacció i el rendiment de transaccions, sense renunciar a la descentralització o la seguretat. Si bé la velocitat i el rendiment són importants, és fonamental que les solucions d'escalat que permetin aquests objectius es mantinguin descentralitzades i segures.

Es classifica l'escalat com a escalat a cadena o escalat fora de cadena:

- **Escalat en cadena:** requereix canvis de protocol Ethereum, s'esperava que la fragmentació de la cadena de blocs escalés Ethereum. Això implicaria dividir la cadena de blocs en fragments per ser verificades per subconjunts de validadors (Sharding). Els subconjunts de validadors serien responsables dels fragments individuals en lloc de fer un seguiment de tot Ethereum. Sharding va estar al full de ruta d'Ethereum durant molt de temps, però el desenvolupament de la capa 2 va fer que la comunitat preferís l'escalat centrat en el rollup en lloc del sharding.

- **Escalat fora de cadena:** les solucions s'implementen per separat de la xarxa principal de la capa 1: no requereixen canvis en el protocol existent. Comencem amb les solucions de capa 2. Aquesta categoria de solucions fora de la cadena deriva la seva seguretat de Mainnet Ethereum.

  És un terme per ajudar a escalar la vostra aplicació mitjançant el maneig de transaccions des de la xarxa principal d'Ethereum (capa 1) mentre s'aprofita el model de seguretat descentralitzat de Mainnet. La velocitat de la transacció es satura quan la xarxa està ocupada, fent que l'experiència de l'usuari sigui dolenta per a certs tipus d'aplicacions. I a mesura que la xarxa es fa més transitada, els preus del gas augmenten a mesura que els remitents de les

transaccions volen superar-se els uns als altres. Això pot fer que utilitzar Ethereum sigui molt car.

La majoria de les solucions de la capa 2 es centren al voltant d'un servidor o grup de servidors, cadascun dels quals pot ser referit a com un node. Depenent de la implementació, aquests nodes de capa 2 poden ser gestionats pels individus, empreses o entitats que els utilitzen (similar a Mainnet). En termes generals, les transaccions s'envien a aquests nodes de la capa 2 en lloc de ser enviades directament a la capa 1 (Mainnet). Per a algunes solucions, la instància de la capa 2 els agrupa en grups abans d'ancorar-los a la capa 1, després de la qual estan assegurats per la capa 1 i no poden ser alterats. Els detalls de com es fa això varien significativament entre les diferents tecnologies de capa 2 i les implementacions.

Per què necessitem la capa 2? L'augment de transaccions per segon millora molt l'experiència de l'usuari i redueix la congestió de la xarxa. Les transaccions s'agrupen en una única transacció a la Mainnet, reduint les tarifes de gas. La solució més important de capa 2 són els rollups.

Un rollup es basa a agrupar diferents transaccions de la xarxa d'Ethereum en una sola. Això es tradueix en el fet que és possible, dins d'un mateix bloc, agrupar una major quantitat de transaccions, fent que s'obtingui una reducció considerable en el pagament per comissió. D'aquesta manera, s'augmentaria la taxa de transacció per segon (TPS), passant de 15 TPS en l'actualitat a més de 1.000.

Aquest rendiment de TPS pot variar segons el rollup que s'implementi. Dins de la recerca realitzada, rollups com Starkware, ZKsync i Loopring, superaven amb escreix les 1.000 TPS, col·locant-se, en un rendiment òptim, al voltant de les 2.000 TPS en mitjana.

Un altre punt analitzat és l'estalvi en gas. Aquest és un dels objectius principals dels rollups. Dins de la recerca no es deixa en clar quin rollup ofereix la millor taxa referent a l'estalvi en gas, però sí que ofereix una comparativa aproximada entre l'estalvi que cada rollup pot arribar a aconseguir, estimant també, la despesa per comissió que comporta utilitzar cada rollup.

Criterio	Zksync	Aztec	Starkware	Loopring	Arbtitrum	Optimism	Fuel
Tipo	ZK-rollup	ZK-rollup	ZK-rollup / Validum	ZK-rollup	Arbitrum	O-RU	O-RU
Disponibilidad de datos	SI	SI	SI	SI	SI	SI	SI
Suposición de watchtower honesta	NO	NO	NO	NO	SI	SI	SI
Suposición de salida masiva	NO	NO	NO	NO	SI	SI	SI
Configuración	CRS	CRS	Transparent	Confianza	no	no	no
Usabilidad	Personalizada	Personalizada	Personalizada	Reparada	Full-EVM	Full-EVM	Full-EVM
Costo de gas en situación optima	~1200	~14k	~300	~450	~2000	~5k-21k	<2000
Costo optimo en situación sub-optima	+++	+++	+++++	++++	+	+	+
Costo fijo (gas)	~500K	~670K	~2,5M-5M	~300K	~36K	ausente	~60K
Tiempo de prueba	2-14'	Desconocido	~3-5'	~7	NO	NO	NO
Costo de prueba	Depende del tiempo de prueba, la compensación con el rendimiento real y los requisitos de capital				NO	NO	NO
Finalidad dificil	Depende del tiempo de prueba				Conpensación con seguridad		
Finalidad suave	Compensación con un costo de gas subóptimo, eficiencia de capital y requerimiento de capital						
Rendimiento máximo	~300	~60	~3000	~1800	~390	desconocido	500
Rendimiento empírico	Depende del tiempo de prueba, la compensación con el costo de la prueba y el requisito de capital				Depende del tamaño del bloque		
Experiencia como Ethereum	NO	NO	NO	NO	SI	SI	SI
Requerimiento de capital	Depende de la finalidad blanda, la finalidad dura y el rendimiento empírico						
Capital	Depende de la finalidad blanda, la finalidad dura						

Figura 28: comparativa dels diferents rollups de Kyber Network [53]

Existeixen dos tipus de rollups:

- **Rollups optimistes:** protocols que augmenten la producció de transaccions en agrupar múltiples transaccions en lots que es processen fora de la cadena. Després, les dades de transacció es registren en la cadena principal amb tècniques de compressió de dades que ajuden a baixar els costos i augmentar la velocitat. Segons Ethereum, els rollups optimistes poden millorar l'escalabilitat de 10 a 100 vegades.

Les transaccions són vàlides per defecte per augmentar l'eficiència. Fan servir un esquema a prova de fraus, amb un termini de resolució de disputes conegut com a "període de desafiament". Dins d'aquest termini, qualsevol persona que monitori el rollup pot enviar un desafiament per a verificar si la transacció es va processar de manera precisa a través d'una prova de frau. Si es descobreixen errors, el protocol de rollup els rectificarà en tronar a executar les transaccions errònies i actualitzar el bloc. Es penalitzaran a les parts que aproven transaccions incorrectes per a la seva execució.

Encara que no hi hagi el procés de validació de les transferències, hi ha el període de desafiament, cosa que augmenta el temps perquè es compleixin aquestes.

Com funcionen els rollups optimistes? Els usuaris envien transaccions als "operadors", que són nodes responsables de processar les transaccions en el resum optimista. També conegut com a "validador", l'operador agrega transaccions, comprimeix les dades subjacents i publica el bloc a Ethereum.

Tot i que qualsevol persona pot convertir-se en validador, els validadors optimistes han de proporcionar un vincle abans de produir blocs, com un sistema de prova de participació. Aquest vincle es pot reduir si el validador publica un bloc no vàlid o es basa en un bloc antic però no vàlid (encara que el seu bloc sigui vàlid). D'aquesta manera, les acumulacions optimistes utilitzen incentius per garantir que els validadors actuen amb honestedat.

S'espera que altres validadors de la cadena de rollup optimista executin les transaccions enviades utilitzant la seva còpia de l'estat del rollup. Si l'estat final d'un validador és diferent de l'estat proposat per l'operador, poden iniciar un desafiament i calcular una prova de frau.

Algunes persones també consideren que els rollups optimistes són menys eficients que els zkRoll-ups. Amb els rollups optimistes, les dades de totes les transaccions s'han de publicar en la cadena per a completar les transaccions.

- **Rollup de zero coneixement (zkRoll-up):** agrupen transaccions perquè s'executin fora de la cadena principal. Per a cada lot, un operador de ZK-rollup enviarà un resum dels canvis requerits una vegada que les transaccions s'hagin executat. Els operadors tenen una funció addicional en la producció de proves de validesa per a provar que els canvis són exactes. Aquestes proves són significativament més petites que les dades de transaccions. Per tant, verificar-les és més ràpid i més econòmic.

En Ethereum, els ZK-rollup redueixen les dades de transacció mitjançant tècniques de compressió quan es registren les transaccions en Ethereum, la qual cosa redueix efectivament les comissions dels usuaris.

L'estat del ZK-rollup es manté mitjançant un contracte intel·ligent desplegat a la xarxa Ethereum. Per actualitzar aquest estat, els nodes ZK-rollup han d'enviar una prova de validesa per a la verificació. Com s'ha esmentat, la prova de validesa és una garantia criptogràfica que el canvi d'estat proposat pel conjunt és realment el resultat de l'execució del grup de transaccions donat. Això vol dir que els ZK-rollups només han de proporcionar proves de validesa per finalitzar les transaccions a Ethereum

en lloc de publicar totes les dades de transaccions a la cadena com a acumulacions optimistes.

No hi ha retards a l'hora de moure fons d'un ZK-rollup a Ethereum perquè les transaccions de sortida s'executen un cop el contracte ZK-rollup verifica la prova de validesa. Per contra, la retirada de fons de les acumulacions optimistes està subjecta a un retard per permetre que qualsevol persona pugui desafiar la transacció de sortida amb una prova de frau.

Com funcionen els rollups de zero coneixement? Els usuaris del ZK-rollup signen transaccions i s'envien als operadors de L2 per al seu processament. En alguns casos, l'operador és una entitat centralitzada, anomenada seqüenciador, que executa transaccions, les agrega en lots i les envia a L1. El seqüenciador d'aquest sistema és l'única entitat autoritzada per produir blocs L2 i afegir transaccions acumulatives al contracte d'acumulació ZK.

Altres ZK-rollups poden rotar el rol d'operador mitjançant un conjunt de validadors de prova de participació . Els possibles operadors dipositen fons al contracte acumulat, amb la mida de cada participació que influeix en les possibilitats de l'apostador de ser seleccionat per produir el següent grup de transaccions. La participació de l'operador es pot reduir si actua de manera maliciosa, la qual cosa l'incentiva a publicar blocs vàlids.

	Rollup optimista	**Rollup zero coneixement**
Validació	Per defecte	Criptografia
Retirar fons	Retard pel període de desafiament.	Quan es verifica la prova de validació
Prova de validació	No son publicats a la cadena	Si son publicats

Taula 6: diferències entre els dos rollups.

Encara que ens trobem amb certes limitacions, la programabilitat d'Ethereum continua sent una eina important amb ple desenvolupament i creació d'una àmplia gamma d'aplicacions descentralitzades, tokens i solucions financeres. Amb les actualitzacions en la cadena de blocs i constant innovacions s'espera que es vagin resolent els problemes o es redueixin el màxim possible.

6.3 Diferències entre Bitcoin i Ethereum

Per a acabar l'apartat, es resumirà les diferències entre Bitcoin Script i Solidity, els llenguatges de programació utilitzats en Bitcoin i Ethereum respectivament.

Bitcoin Script és un llenguatge Touring incomplet, limitat dissenyat per a executar operacions simples en la xarxa Bitcoin, principalment per a crear transaccions bàsiques. En Bitcoin, s'utilitza per a establir condicions que han de complir-se per a gastar fons.

D'altra banda, Solidity és un llenguatge de programació Turing complet utilitzat en Ethereum per a crear smart contracts. Aquests contractes intel·ligents són programes autònoms que s'executen en la cadena de blocs i permeten una àmplia gamma d'aplicacions descentralitzades, des de finances fins a jocs i més.

Bitcoin Script s'enfoca en la seguretat i la simplicitat, mentre que Solidity ofereix flexibilitat i automatització avançada. Mentre que Bitcoin Script és utilitzat per a operacions de transacció, Solidity permet la creació d'aplicacions descentralitzades complexes amb interaccions programables entre participants i actius digitals.

Per a abordar això, tant a Bitcoin com a Ethereum s'està implementant solucions de capa 2, per a millorar el rendiment fora de la cadena principal, i d'aquesta manera millorar en escalabilitat, programabilitat i rapidesa.

En conjunt, aquestes diferències ressalten els rols únics que juguen Bitcoin Script i Solidity en les seves respectives plataformes, brindant funcionalitats essencials per a transaccions i aplicacions descentralitzades en les seves respectives cadenes de blocs.

	Bitcoin Script	Solidity
Funcionalitat	Limitada	Touring Complet
Ús principal	Crear transaccions	Smart contracts
Complexitat	Senzilla	Avançada
Apps	Transaccions, bloqueig de fons	Apps descentralitzades
Flexibilitat	Limitada	Àmplia
Seguretat	Elevada	Prioritària però flexible
Casos d'ús	Transaccions	Finances descentralitzades, jocs...
Solucions capa 2	Lighting Network, Rootstock	Optimistic Rollup, zk-Rollups

Taula 7: Diferències entre Bitcoin Script i Solidity

7 Desenvolupament d'una dApp

En aquest darrer punt s'estudia i s'implementa una aplicació descentralitzada senzilla basada en un sistema de votació electoral, per tal de veure el funcionament de la cadena de blocs. Es va arribar a aquesta idea, ja que amb tots els avantatges que ens presenta la blockchain aquest ús podria ser usat més endavant en les eleccions dels països per tal de mantenir un anonimat per part dels votants i evitar trampes o modificacions en els resultats.

Aquesta aplicació seria una primera mostra del que hauria de ser una de gran escala, aquesta ha de ser capaç de realitzar vots i mantenir un recompte en temps real dels partits polítics existents. A continuació es detallaran les eines que s'han fet servir en la implementació de la dApp i la seva estructura, així com el procés que segueix quan es produeix un vot.

7.1 Tecnologies utilitzades

Per a la implementació de la dApp s'utilitzaran les següents tecnologies:

- **Ionic:** es tracta d'un SDK de codi obert per a desenvolupar aplicacions mòbils híbrides, tant per a Android, IOS, Windows, etc. aquest framework està construït sobre Angular, Vue i React permetent l'ús de CSS, HTML, i TS per al desenvolupament del codi.

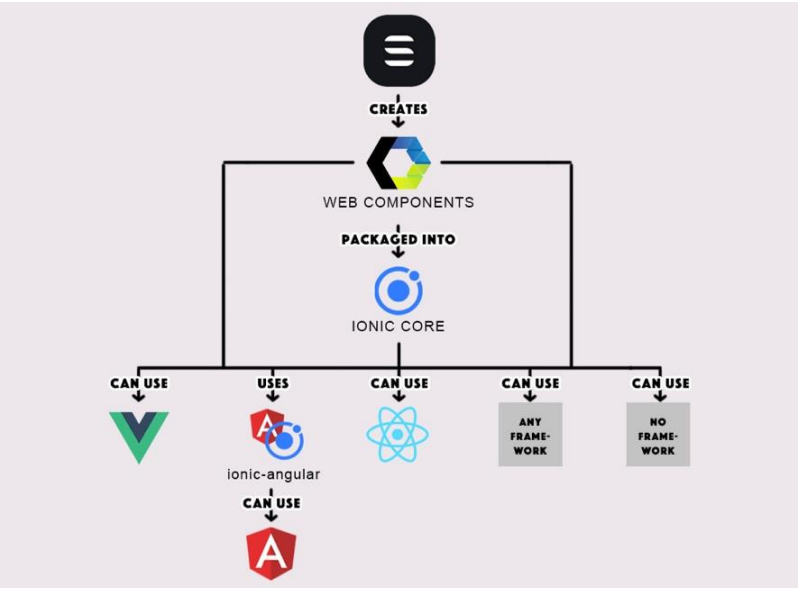


Figura 29: arquitectura Ionic [58]

- **React:** és una plataforma de desenvolupament de codi obert creada per Facebook que es fa servir per a crear aplicacions mòbils per a iOS i Android. Es basa en el popular marc de treball React, que s'utilitza per al desenvolupament d'aplicacions web. React Native permet als desenvolupadors escriure codi en JavaScript i React per crear aplicacions mòbils que es comportin de manera nativa en els dispositius.

  Es pot reutilitzar una gran part del codi per a les dues plataformes (iOS i Android). Les aplicacions React Native tenen un aspecte i un rendiment natius. A més, hi ha una gran comunitat de desenvolupadors i molts components i llibreries disponibles.

- **Hardhat:** conjunt d'eines i un marc de desenvolupament per a la creació i el desplegament de contractes intel·ligents (smart contracts) en la plataforma Ethereum i altres blockchains compatibles amb Ethereum.

  Aquesta eina facilita el desenvolupament, la prova i la gestió de projectes de contractes intel·ligents, i és àmpliament utilitzada pels desenvolupadors d'aplicacions descentralitzades i altres solucions basades en Ethereum.

- **Ganache:** eina de desenvolupament utilitzada en l'ecosistema d'Ethereum per a crear i gestionar una cadena de blocs local o una xarxa de prova personalitzada. És una de les eines més populars per a desenvolupadors que treballen en projectes relacionats amb Ethereum, incloent-hi el desenvolupament i prova de contractes intel·ligents i aplicacions descentralitzades.

  Ens permetrà implementar l'app en un entorn segur i de testatge pel nostre smart contract, de fet, és qui ens permet realitzar les transaccions de test, ja que disposa de comptes amb quantitats grans d'ethers.

- **Metamask:** és una cartera de criptomonedes i una extensió de navegador que permet als usuaris gestionar la seva identitat digital, emmagatzemar i enviar ether (la criptomoneda nativa d' Ethereum) i tokens ERC-20, i interactuar amb aplicacions descentralitzades en la xarxa Ethereum.

Amb aquesta cartera, permetrà a l'usuari iniciar sessió amb el seu compte Ethereum per poder realitzar el vot. Hi ha altres opcions, però aquest cas ens facilita la feina amb la seva API per a implementacions.

- **Visual Studio Code:** és un editor de codi lleuger però molt potent. Ve amb suport instal·lat per a JavaScript, Typescript i Node.js i se li poden afegir una infinitat d'extensions de molta utilitat i és una eina de codi obert.

  En el cas d'aquest projecte, s'han utilitzat diferents extensions proporcionades per aquest IDE, com per exemple Typescript, HTML, CSS pel desenvolupament del FrontEnd de les aplicacions o JavaScript pel desenvolupament de la part de Blockchain.

- **Github:** Es tracta d'una plataforma dissenyada per a albergar projectes utilitzant un sistema de control de versions Git, la qual cosa es complementa a la perfecció amb l'eina comentada anteriorment. Ofereix possibilitat d'afegir arxius per a fer més visuals els projectes i les seves definicions. Es poden observar també gràfics del desenvolupament pels diferents col·laboradors, així com les seves branques de treball.

7.2 Estructura de la Dapp

En aquest punt es mostrarà quina és l'estructura del projecte i s'explicarà aquelles carpetes i fitxers més importants.

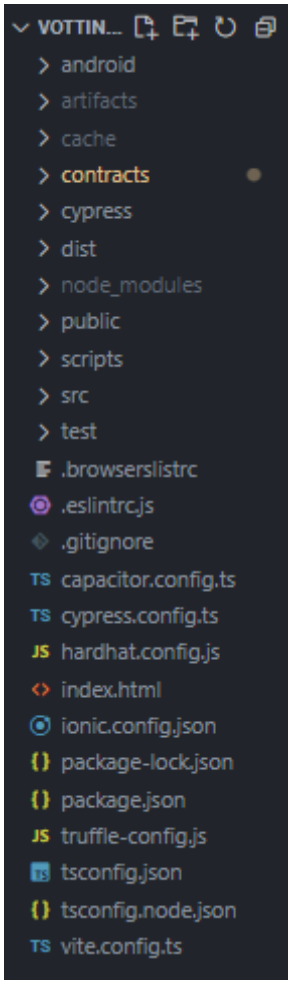


Figura 30: estructura del projecte

S'observa com hi ha una quantitat considerada de fitxers i carpetes, ja que es tracta d'un projecte on s'ha de dividir cadascuna de les parts que corresponen a una funció o una altra.

Començar amb la carpeta **android**, on es troben tots els fitxers de configuració encarregats que l'aplicació funcioni en un dispositiu android, cosa que també funcionaria si es volgués en ios. Aquesta carpeta es genera un cop s'hagi executat la següent comanda:

Ionic capacitor build android/ios

Per altra banda, tenim la carpeta artifacts, generada per hardhat quan es compila un contracte. El fitxer més important, però és el “nomContracte”.json perquè conté l’ABI (Application Binary Interface), les funcions i les dades del contracte codificades per a poder-se comunicar amb l’entorn blockchain. L'ABI proporciona una descripció formal de les funcions del contracte intel·ligent, incloent-hi els tipus de paràmetres que accepten, els tipus de dades que retornen i com s'han d'estructurar les dades per cridar a aquestes funcions.

Figura 31: part d’ABI on es mostra dos inputs del contracte

Seguidament destacar la carpeta contract. Potser la part més important del projecte, ja que s'implementa l'smart contract de l'aplicació. On es detalla les funcions de la dApp per tal de poder interactuar amb la blockchain.

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.9;


contract VotingContract {

    struct Candidate {
        uint id;
        string name;
        string politic;
        uint voteCount;
    }

    address owner;
    Candidate[] public candidates;
    mapping(address => bool) public hasVoted;

    uint public candidatesCount = 0;

    //constructor with an array of candidates
    constructor(string[] memory _candidateNames, string[] memory _politics) {
    for (uint i = 0; i < _candidateNames.length; i++) {
        candidates.push(Candidate({
            id: candidatesCount,
            name: _candidateNames[i],
            politic: _politics[i],
            voteCount: 0
        }));

       candidatesCount++;
    }
    owner = msg.sender;

    }

    // register a vote
    function vote(uint256 _candidateIndex) public {

        candidates[_candidateIndex].voteCount++;
        hasVoted[msg.sender] = true;
    }

    function getAllVotesOfCandiates() public view returns (Candidate[] memory){
        return candidates;
    }


}
```

Figura 32: smart contract del projecte

En la figura anterior es mostra el contracte intel·ligent, observem que primer es declara la versió de Solidity amb la qual s'ha de compilar el contracte. Després es defineix l'estructura del candidat i les variables d'estat: **owner**, que emmagatzema l'adreça de l'usuari que crea el contracte, **candidates**, un array pública dels candidats que participen, **hasVoted**, és un mapping que manté el registre dels usuaris que han votat, i **candidateCount**, l'índex i comptador dels candidats.

El constructor s'executa una sola vegada quan es crea el contracte. Accepta dues cadenes com a paràmetres: noms dels candidats i els seus partits polítics. A continuació, crea els candidats utilitzant aquesta informació i inicialitza les seves propietats.

La funció **vote,** permet als usuaris realitzar el vot per un candidat específic. Incrementa el compte de vots del candidat seleccionat i marca l'usuari com a "hasVoted" per evitar vots duplicats. En aquest cas, però es comenta la línia que evita el duplicat per tal de mostrar en l'aplicació el funcionament correcte.

Per acabar, la funció **getAllVotesOfCandidates**, permet als usuaris consultar tots els candidats i els vots que han rebut.

Seguint en la carpeta script es troben aquells fitxers encarregats de fer el deploy dels contractes intel·ligents. En aquest cas, tenim un fitxer anomenat **deploy.js** on es crea el contracte amb els dos arguments d'entrada que s'han comentat anteriorment. A la part superior hi ha l'array corresponent a tots els candidats.

```
const cands = [ ···
];

//funcion main fa el deploy del contracte

async function main() {
    const candidateNames = [];
    const candidatePolitic = [];

    //afageix els noms i partits politics pel constructor
    for (const candidate of cands) {
        candidateNames.push(candidate.name);
        candidatePolitic.push(candidate.politic);
    }

    const [deployer] = await ethers.getSigners();

    const VotingContract = await ethers.getContractFactory("VotingContract");
    const contract = await VotingContract.deploy(
        candidateNames,
        candidatePolitic
    );
}

main()
    .then(() => process.exit(0))
    .catch((error) => {
        console.error(error);
        process.exit(1);
    });
```

Figura 33: script per fer el deploy del contracte

Arribem a la carpeta més important, **src,** aquí es troba tot el frontend de l'aplicació, és a dir, components i pàgines.

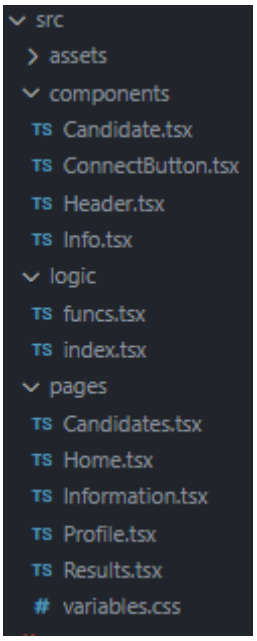


Figura 34: interior de la carpeta source

En la primera, components s'observa:

- **Candidate**: mostra cadascuna de les cartes dels candidats, ensenyant així nom i partit polític per a l'usuari.

- **ConnectButton**: encarregat de connectar l'aplicació amb la cartera de metamask, si no és així, mostra una icona per entrar al perfil de l'usuari.

- **Header**: sempre està fix en l'aplicació, mostra el títol i si l'usuari està connectat el botó per entrar al perfil, sinó el botó per connectar Metamask.

- **Info**: quan es clica el component de candidate, s'obre un modal amb més informació detallada del partit polític seleccionat. Cosa que facilita per a realitzar el vot.

Per altra banda, parlant de les pàgines tenim:

- **Candidates**: mostra tots els candidats en graella, és a dir, files i columnes que s'adapta a la mida de la pantalla de manera recursiva.

- **Home**: primera pàgina de l'aplicació, per donar la benvinguda a l'usuari amb títol i una breu descripció de les eleccions. Cal destacar que sempre hi ha un "footer", per tal de tenir una navegació còmoda.

- **Information:** encarregat d'ensenyar a l'usuari un punt d'informació sobre les eleccions, cosa que li dona una ajuda a l'usuari.

- **Profile**: ensenya detalls del compte de Metamask com número de compte, el balanç d'ether disponibles...

- **Results:** en aquesta pàgina s'observa els resultats en viu de les eleccions, agafant de la blockchain els vots i actualitzant-los en el moment.

El fitxer encarregat de les funcions que permeten la interacció amb el contracte intel·ligent es troba dins de la carpeta logic, **funcs.tsx.**

```
import VotingContractABI from "../../artifacts/contracts/VotingContract.sol/VotingContract.json"; // Importa el ABI del contracte
import { ethers } from "ethers";

//direccio del contracte creat, canviar quan es desplegui un de nou
const contractAddress = "0x85461EeD54Cf2741BD290396297b2af1d163009b";

export const cands = [···
];

//funcio per a realitzar el vot a un candidat especific
export async function vote(id: any) {
    const provider = new ethers.BrowserProvider(window.ethereum);
    const signer = await provider.getSigner();
    const contract = new ethers.Contract(
        contractAddress,
        VotingContractABI.abi,
        signer
    );

    const tx = await contract.vote(id);
    await tx.wait();

    getCandidates();
}

//retorna els candidats de les eleccions
export async function getCandidates() {
    const provider = new ethers.BrowserProvider(window.ethereum);
    const signer = await provider.getSigner();
    const contract = new ethers.Contract(
        contractAddress,
        VotingContractABI.abi,
        signer
    );
    const candidatesList = await contract.getAllVotesOfCandiates();
    const formatCandidates = candidatesList.map(
        (candidate: any, index: any) => {
            return {
                index: index,
                name: candidate.name,
                politic: candidate.politic,
                voteCount: Number(candidate.voteCount.toString()),
            };
        }
    );
    return formatCandidates;
}
```

Figura 35: fitxer amb les funcions d'interacció amb smart contract

La primera cosa a comentar és l'adreça del contracte, sense aquesta, no es fa res. Un cop s'ha fet el deploy, s'ha de copiar l'adreça i pegar-li en aquesta variable per tal que el funcionament sigui el correcte, si no, no es podrà connectar el frontend amb el contracte.

Després s'implementen dues funcions: **vote()** i **getCandidates()**. Totes dues pertanyen a les del contracte intel·ligent, però en aquest fitxer es realitza la connexió amb el contracte.

Aquí com que aconseguim el provider (nterfície per connectar-se a la xarxa i interactuar amb els seus nodes i smart contracts) i el signer (component que té la capacitat de signar digitalment dades o transaccions amb una clau privada.) perquè ens trobem connectats a l'aplicació, juntament amb l'ABI i l'adreça del contracte ja tenim accés per poder executar les funcions implementades en el contracte.

Per acabar, a destacar de l'estructura és **hardhat.config.js** on es situa la clau privada i l'URL de la xarxa de Ganache, cosa que ens permet connectar-lo amb aquest, l'aplicació i Metamask.

7.3 Resultat final

Aquest apartat es dedicarà a fer un recorregut de tota l'aplicació per ensenyar el resultat final.

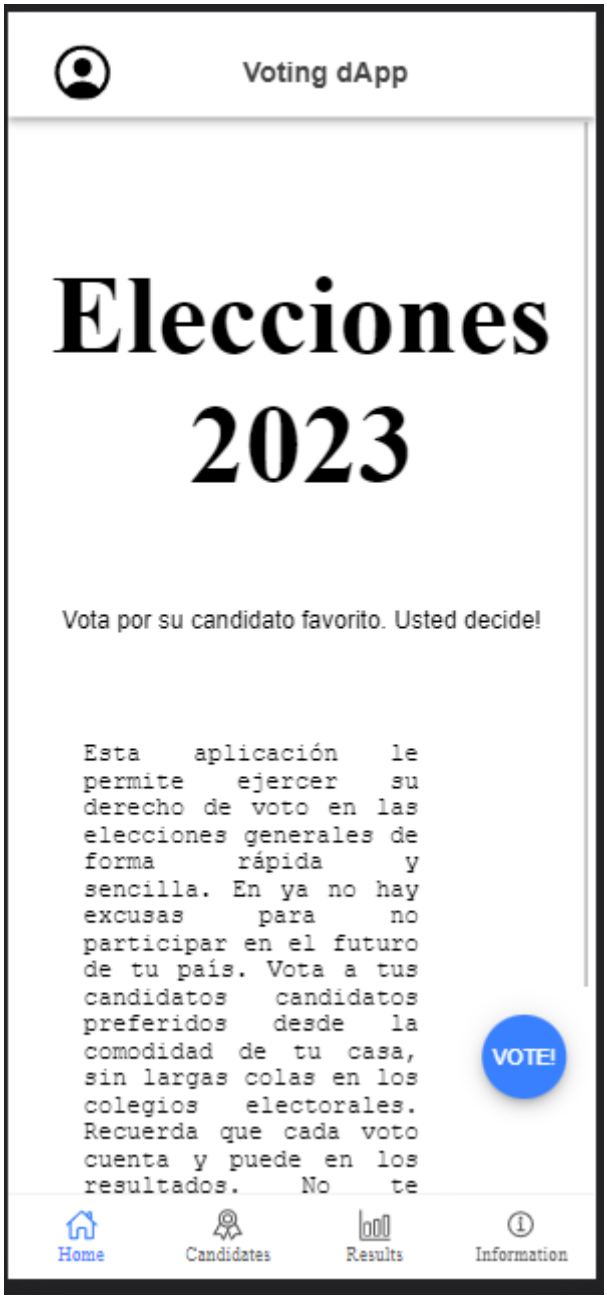
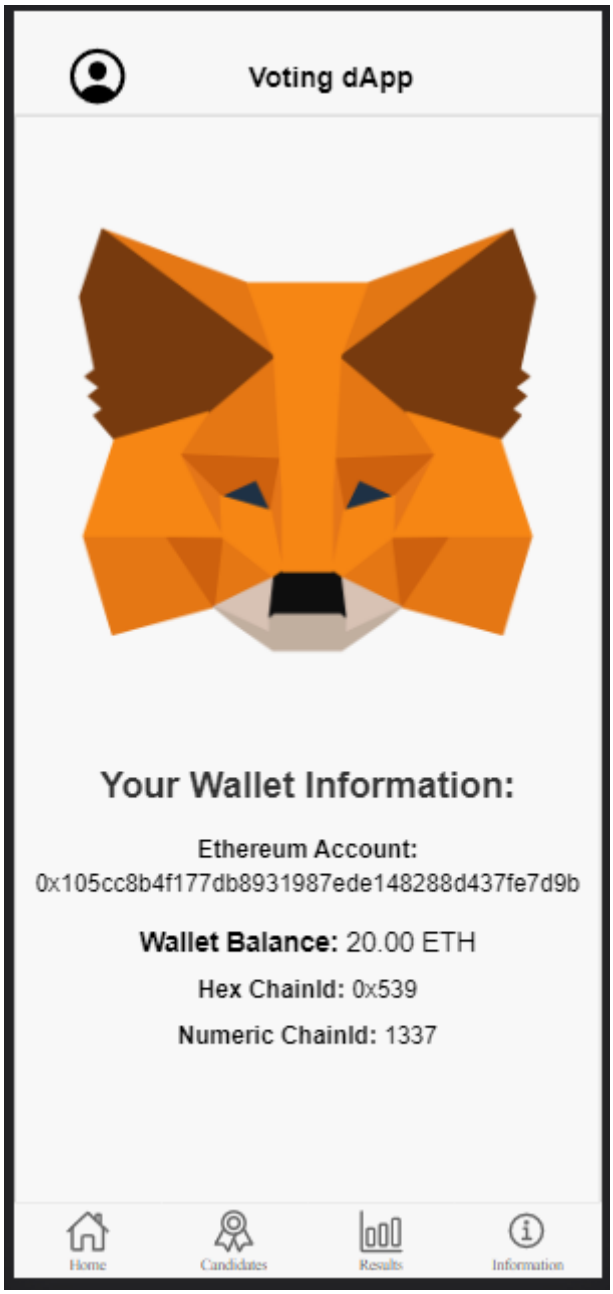


Figura 36: pàgina d'inici

Pàgina d'inici, observem l'estructura principal del projecte amb les quatre pestanyes a sota per facilitar la navegació, de fons tenim el títol amb la descripció de les eleccions. Destacar el botó flotant per anar directe a la votació i el header amb el títol de l'aplicació i el botó encarregat de portar-nos al perfil.



Com s'ha comentat anteriorment, mostra la informació del nostre compte connectat a l'aplicació. Número de compte, saldo disponible d'ethers i número de la cadena.

Figura 37: pàgina de perifl

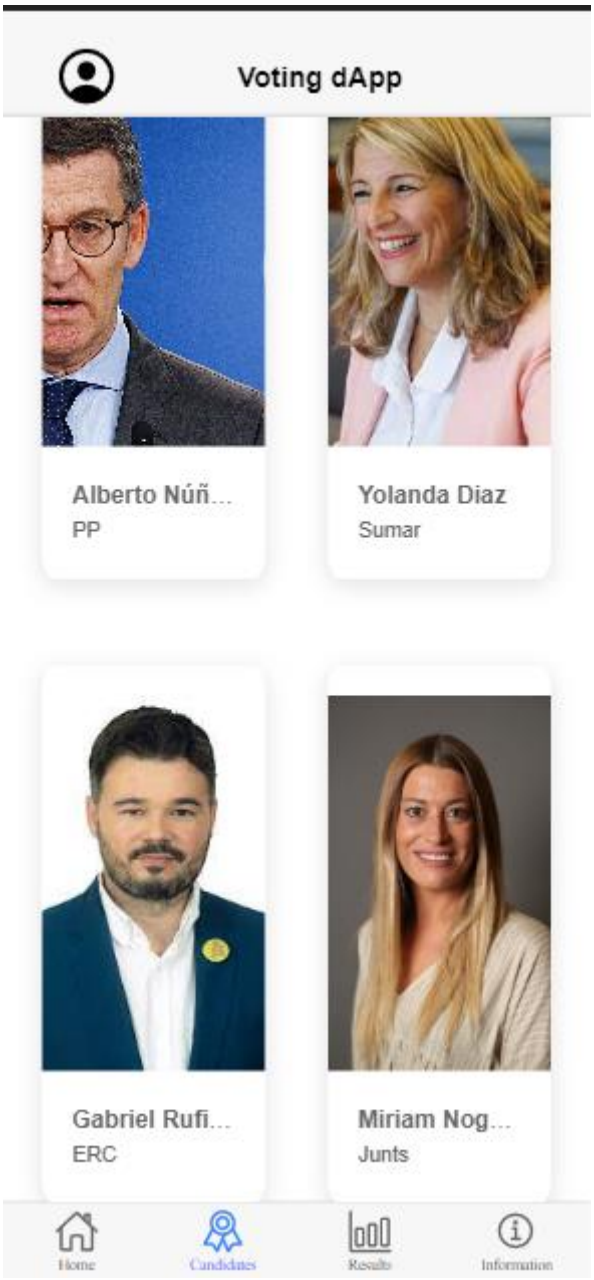
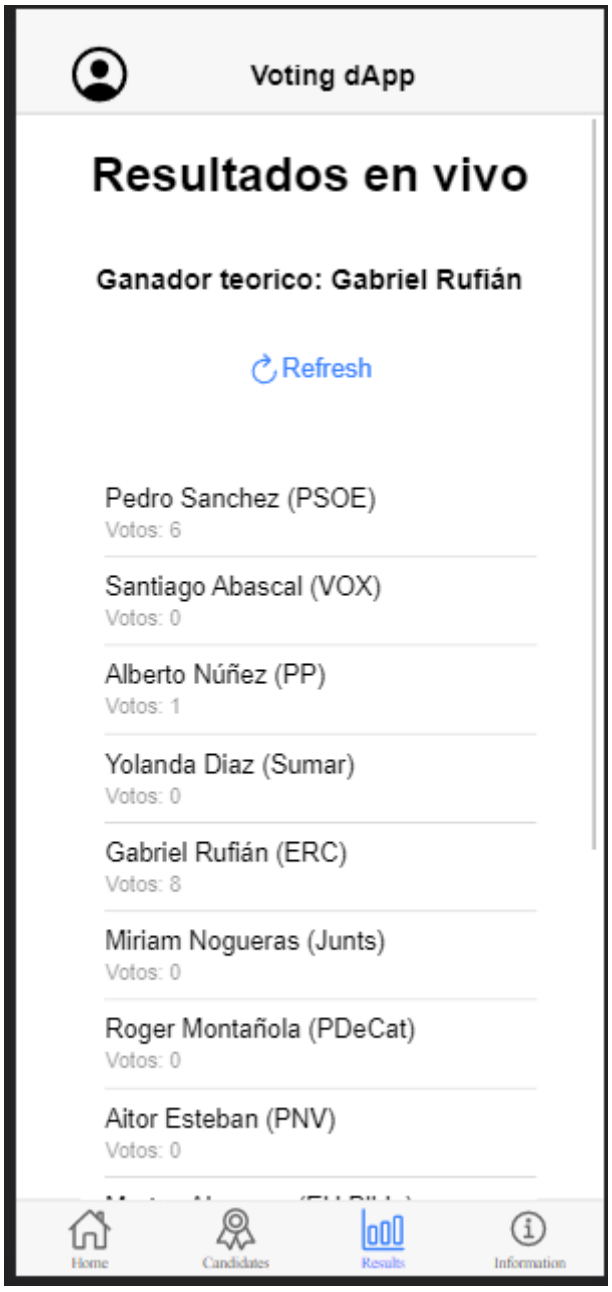


Figura 38: pàgina amb la graella de candidats

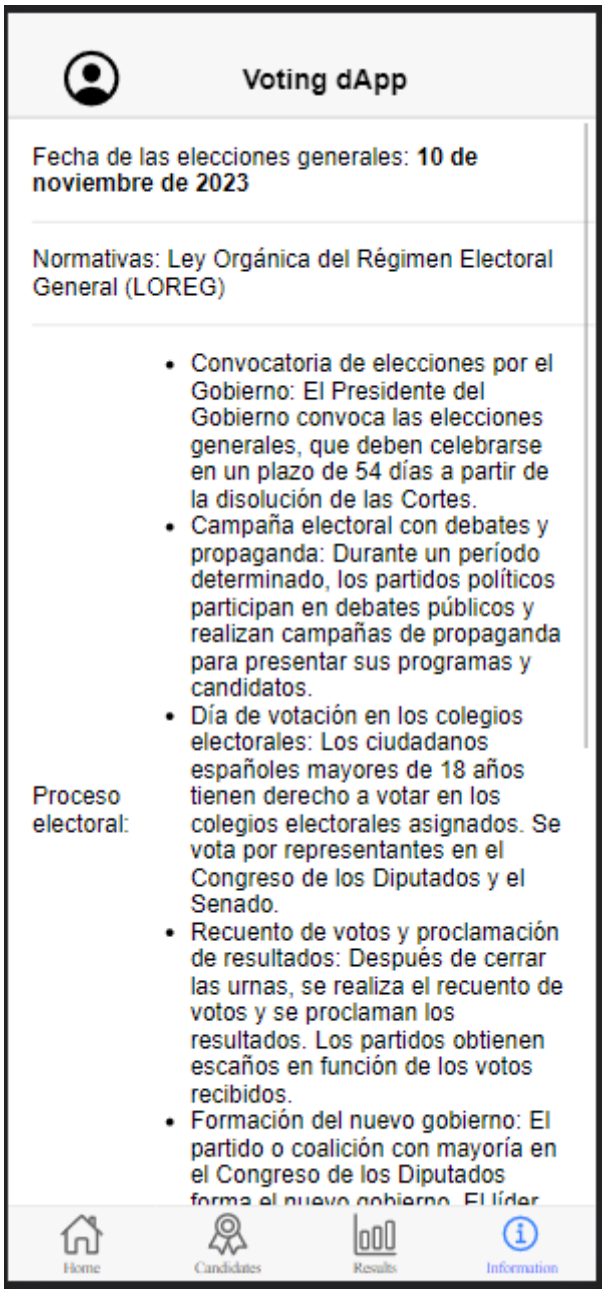
Figura 39: modal del partit polític

S'observen tots els candidats situats en files i columnes, cada carta és un botó que en polsar-lo, s'obrirà un modal (com mostra la figura del costat) per acabar de fer el vot, a part d'ensenyar una descripció de cada partit polític.



La pàgina ensenya els vots de cada partit polític en forma de llista, sense estar ordenats. Hi ha un botó de refrescar per tal d'actualitzar la llista i el guanyador teòric en cas que no ho hagi fet.

Figura 40: resultats de les eleccions



Última pestanya de navegació, ens informa (situació hipotètica), normes i requisits a seguir de les eleccions.

Figura 41: pàgina d'informació electoral

7.4 Explicació del procés i canvi realitzat a la blockchain

El procés de realitzar un vot a través de la blockchain és una operació que implica diversos passos i participants. Aquí es detalla una explicació del que succeeix:

L'usuari que vol votar ha d'estar en possessió d'una cartera de criptomonedes que contingui una quantitat de la criptomoneda nativa de la xarxa blockchain en la qual es realitzarà la votació. Aquesta criptomoneda s'utilitzarà per pagar les tarifes de la transacció. En aquest cas, la cartera serà Metamask i la comenda ether.

A més, ha de tenir una connexió a la xarxa blockchain a través d'una aplicació o cartera de criptomonedes que li permeti interactuar amb la xarxa. Aquesta aplicació fa servir un provider per connectar-se a la xarxa, en aquesta aplicació es fa servir Ganache.

L'usuari selecciona el candidat pel qual vol votar a través de la interfície de l'aplicació. Aquesta selecció es tradueix en dades que seran incloses en la transacció. La cartera de l'usuari crea una transacció que contingui les següents dades:

- L'adreça del candidat seleccionat.
- La quantitat de criptomoneda que es farà servir per pagar les tarifes de la transacció.
- Un camp de dades que contingui informació específica del vot.

L'aplicació usa el signer de l'usuari per signar digitalment la transacció. Això es fa utilitzant la clau privada associada amb l'adreça de l'usuari. Metamask en aquest procés, ens mostrarà un avís per tal de confirmar la transacció. Seguidament, l'usuari envia la transacció signada a la xarxa blockchain.

La transacció s'envia a través de la xarxa i arriba a diversos nodes que verifiquen la signatura digital i la integritat de la transacció. Un miner de la xarxa selecciona la transacció i l'inclou en un bloc que formarà part de la cadena de blocs.

La transacció és confirmada quan el bloc que la conté és afegit a la cadena de blocs. El nombre de confirmacions pot variar segons la xarxa, però cada confirmació fa que la transacció sigui més segura.

Finalment, el vot es registra a la cadena de blocs de manera immutable i queda registrat permanentment. La plataforma de votació actualitza el recompte de vots i mostra els resultats actualitzats als usuaris.

En resum, el procés de votació mitjançant la cadena de blocs garanteix la seguretat i la transparència de les eleccions, ja que tots els vots es registren de manera permanent i no es poden modificar ni manipular un cop a la cadena de blocs. D'aquesta manera, proporciona una solució segura per a les eleccions i altres processos de votació.

El github de l'aplicació és el següent, s'ha de canviar a la branca master:

https://github.com/roviralta/vottingApp

8 Conclusions

El principal objectiu d'aquest projecte era investigar, experimentar i aprendre sobre la tecnologia de la cadena de blocs, destacant Bitcoin i Ethereum. I aquest objectiu s'ha aconseguit mitjançant un estudi teòric profunditzant aspectes tècnics i funcionament de les dues tecnologies, una anàlisi de l'estat d'art destacant la seva programabilitat i finalitzant amb la implementació d'una petita aplicació descentralitzada basada en un sistema de votació.

Gràcies a la realització del projecte el resultat ha estat l'esperat permetent veure que és una tecnologia amb un gran potencial, que segur que cada any anirà creixent i apareixent en diversos sectors i proporcionant serveis.

A causa del desenvolupament de l'aplicació amb Ethereum ens ha permès conèixer el llenguatge Solidity, corresponent als smart contracts, aprenent així des de zero com es programa, es compila i s'executa en un entorn de test. A més de poder

En resum, important com la tecnologia ha revolucionat la forma com es tracten les dades i transaccions, a més de poder-se utilitzar de manera efectiva i evitant la seva modificació. S'ha aconseguit una aplicació web funcional que compleix els objectius plantejats i que pot servir de base per a futurs projectes orientats a l'ús per un públic de veritat.

9 Bibliografia

[1] https://carballar.com/historia-de-blockchain

[2] https://101blockchains.com/es/historia-de-la-blockchain/

[3] https://www.geeksforgeeks.org/smart-contracts-in-blockchain/

[4] https://academy.bit2me.com/que-es-una-red-p2p/

[5] https://profile.es/blog/que-es-blockchain-fundamentos-basicos-de-la-cadena-de-bloques/

[6] https://academy.bit2me.com/cuantos-tipos-de-blockchain-hay/

[7] https://www.iebschool.com/blog/blockchain-cadena-bloques-revoluciona-sector-financiero-finanzas/

[8] https://es.cointelegraph.com/news/cuantos-algoritmos-de-consenso-existen-para-las-blockchain

[9] https://phemex.com/es/academy/la-criptografia-blockchain

[10] https://coinmarketcap.com/alexandria/glossary/sha-256

[11] https://keccak.team/keccak.html

[12] https://seguridaddigitalvenezuela.blogspot.com/2018/02/que-son-las-funciones-hash-en.html

[13] https://es.cointelegraph.com/learn/what-is-ethereum-a-beginners-guide-to-eth-cryptocurrency

[14] https://www.plus500.com/es-ES/Instruments/ETHUSD/The-History-of-Ethereum~4

[15] https://academy.bit2me.com/que-es-ethereum-virtual-machine-evm/

[16] https://www.bbva.com/es/innovacion/ether-criptomoneda-y-pieza-clave-de-las-finanzas-descentralizadas/

[17] https://academy.bit2me.com/que-es-gas-en-ethereum/

[18] https://ethereum.org/es/developers/docs/evm/

[19] https://es.cointelegraph.com/news/what-is-an-ethereum-virtual-machine-evm-and-how-does-it-work

[20] https://coinacademy.es/guias/como-calcular-el-coste-de-una-transaccion-ethereum-erc20-en-dolares/

[21] https://ethereum.org/es/developers/docs/transactions/

[22]https://etherscan.io/tx/0x0c0ed7f13f0c9fa8c75267d589a55d792e4954399c552ee47d8baf2bb95752a6

[23] https://academy.bit2me.com/diferencias-entre-bitcoin-y-ethereum/

[24] https://www.investopedia.com/tech/what-happens-bitcoin-after-21-million-mined/

[25] https://etherscan.io/

[26] https://academy.bit2me.com/que-son-los-smart-contracts/

[27] https://revistas.ubp.edu.ar/index.php/derecho-notarial-registral/article/view/2362-3845%282019%29002/133

[28] https://www.mailteck.com/smart-contracts-la-guia-definitiva-para-principiantes/

[29] https://observatorioblockchain.com/hypernifty/smart-contracts-que-son-como-funcionan/

[30] https://cointelegraph.com/learn/what-are-smart-contracts-a-beginners-guide-to-automated-agreements

[31] https://academy.bit2me.com/que-es-ipfs/

[32] https://www.slideshare.net/IoannisPsaras/module-content-addressing-in-ipfs

[33] https://academy.bit2me.com/que-es-bitcoin-script/

[34] https://es.cointelegraph.com/explained/what-is-bitcoin-script-the-bitcoin-programming-language

[35] https://www.youtube.com/watch?app=desktop&v=2rTC0a2np1U&ab_channel=LABITCONF

[36] https://academy.bit2me.com/que-son-los-smart-contracts/

[37] https://es.cointelegraph.com/explained/rgb-bitcoin-smart-contracts-and-lightning-network

[38] https://www.rgbfaq.com/

[39] https://blog.bitnovo.com/que-es-el-protocolo-rgb/

[40] https://www.realvision.com/blog/does-bitcoin-have-smart-contracts

[41] https://wiki.lemon.me/bitcoin/que-es-la-escalabilidad-de-bitcoin/#Escalabilidad_de_las_criptomonedas_Bitcoin_no_es_escalable

[42] https://crypto.com/university/es/what-are-bitcoin-layer-2s

[43] https://www.expansion.com/economia-digital/innovacion/2021/11/22/61962546e5fdea61208b45d0.html

[44] https://dcxlearn.com/blockchain/what-is-lightning-network/

[45] https://www.iebschool.com/blog/solidity-lenguaje-programacion-ethereum-tecnologia/

[46] https://www2.deloitte.com/content/dam/Deloitte/us/Documents/financial-services/us-dcfs-blockchain-in-cre-the-future-is-here.pdf

[47] https://www.inmobilia.com/proptech/smart-contracts-en-el-sector-inmobiliario-seguros-confiables/

[48] https://blogthinkbig.com/token-estandares

[49] https://www.researchgate.net/figure/A-portion-of-OpenZeppelins-Solidity-ERC-20-contract-with-snapshots-Its-full-code-is_fig5_364212098

[50] https://www.researchgate.net/figure/A-simplified-portion-of-OpenZeppelins-ERC-721-implementation-in-Solidity-Its-full-code_fig2_364212098

[51] https://academy.bit2me.com/que-es-defi-o-finanzas-descentralizadas/

[52] https://ethereum.org/en/developers/docs/scaling/

[53] https://www.criptonoticias.com/tecnologia/rollups-ethereum-pros-contras-soluciones-congestion-red/

[54] https://academy.binance.com/es/articles/optimistic-vs-zero-knowledge-rollups-what-s-the-difference

[55] https://www.geeksforgeeks.org/introduction-to-rolling-hash-data-structures-and-algorithms/

[56] https://en.wikipedia.org/wiki/Rolling_hash

[57] https://medium.com/algorithm-solving/rolling-hash-ecd7c08358ab

[58] https://www2.deloitte.com/es/es/pages/technology/articles/Ionic-4.html