KAPITULLI 1: HYRJE

Punimi i kësaj diplome do të ketë në fokus implementimin e teknologjive të decentralizimit, sic është blockchain, në lëshimin dhe menaxhimin e pasurive të patundshme, duke filluar me lëshimin e certifikatave të pronësisë. Në fokus do të jetë përmirësimi i procesit aktual, si dhe rritja e integritetit dhe besueshmërisë së certifikatave të pronësisë. Do bëhet një selektim i teknologjive dhe metodikave të punës, si dhe do krijojmë planin kryesor të ndërtimit të rrjetit blockchain dhe komunikimin me përdoruesin. Një projekt me një shkallë kompleksiteti të tillë kërkon një bazë të qëndrueshme mbi të cilën do të lëvizë dhe do të përpunohet informacioni. Duhet marrë parasysh që zgjidhjet e bëra të jenë sa më fleksibël dhe të ketë sa më shumë mundësi manovrimi në rast se dicka shkon keq. Në rastin e blockchain kjo sfidë merr përmasa shumë herë më të mëdha se sistemet tradicionale klient-server-klient. Tek ky i fundit, në rastin më të keq rregullimet kryhen direkt mbi server, ndërsa në një sistem si blockchaini është e pamundur të kryhen modifikime në kontratën virtuale.

1.1. Motivi

Teknologjia blockchain po përdoret akoma dhe më shumë në disa fusha sic janë DeFi (Decentralised Finance), Arsimi, Kriptovaluta, Markete NFT, Ruajtja e plagjatures etj [1]. Por akoma nuk ka një implementim të mirëfilltë në menaxhimin e pasurive të patundshme. Kryesisht pjesa më e madhe synojnë drejt lidhjes pronar-klient apo pronar-investitor, duke automatizuar pjesën e pagesës dhe nevojës së një pale të tretë për zbatimin e marrëveshjeve. Akoma nuk ka një sistem që mbështetet mbi blockchain për vrëtetësinë e një pasurie [2]. Shpesh kemi parë probleme me certifikata të pronësisë apo dokumente të këtij lloji, dhe marrja e një certifikate të thjeshtë është e lodhshme.

KAPITULLI 2: PROBLEMI

Aktualisht, një pjesë e madhe e sistemeve institucionale që kanë të bëjnë me lëshimin e certifikatave, kryesisht bazohen në ruajtjen e të dhënave në një vend të vetëm, të ruajtur nga një entitet, dhe në disa raste aspekti digjital i shërbimeve në fjale është ineksiztent. Për më tepër sistemet aktuale janë të ndjeshme ndaj numrit të kërkësave paralele, si dhe kanë mungesa në verifikimin e integritetit të të dhënave. Një rast ku sistemi do të ishte i papërdorshëm është gjatë një sulmi DDoS, ose humbja e përkohshme (ndoshta edhe e përhershme) e të dhënave si pasojë e infektimeve me viruse dhe sulmeve keqdashëse ndaj sistemit, apo dhe ndaj gabimeve njerëzore [3]. Problem tjetër, sic u përmend më lart, është integriteti i të dhënave dhe mungesa e transparencës, ku kryesisht mungon aksesi digjital i cerfikatës. Shembulli më i thjeshtë është verifikimi i një certifikate pronësie. Kryesisht një certifikatë e tillë jepet në format të printuar, në mënyrë fizike, dhe vuloset sipas standarteve paraprake. Ndonëse e vështirë për tu falsifikuar nga kushdo, një person keqdashës me paisjet e duhura mund të falsifikojë një certifikatë, dhe kjo mund të përsëritet pafundësisht. Edhe pse herët apo vonë falsifikimi mund të kapet, kjo mund të jetë shumë vonë.

KAPITULLI 3: PREZANTIM ME ZGJIDHJEN

3.1. Teknologjia Blockchain

Blockchain është një teknologji relativisht e re, implementimet e së cilës po rriten eksponencialisht dita ditës. Një shpjegim i thjeshtë do të ishte "një databazë e ndarë ndërmjet përdoruesve", ku secili prej anëtarëve të blockchainit ka akses në këtë databazë. Si pasojë e mënyrës si funksionon, kjo teknologji është perfekte për të dhëna të pandryshueshme dhe rekorde të përhershme, ku fokusin kryesor ka decentralizimin. Kjo do të thotë që një implementim i tillë na prezanton me një nivel integriteti të të dhënave që nuk është parë më parë [4].
Megjithëse është një teknologji e cila po implementohet gjerësisht kohët e fundit, kjo bazohet në baza më të hershme të informatikës. Blockchain është një listë rekordesh ose të dhënash të cilat quhen blloqe, dhe lidhen me njëra tjetrën nëpërmjet teknikave kriptografike, ku cdo bllok ka një hash si identifikues, kohën e lëshimit dhe të dhënat. Hash është një seri karakteresh unike për një set të dhënash, dhe në rastin e blockchain cdo hash i një blloku përbëhet nga hashi i bllokut pasardhes. Kjo bën të mundur që në rast modifikimi të blloqeve të mëparshme, cdo bllok pasardhës bëhet i pavlefshëm. Kjo lidhje e hash-eve të blloqeve formon një zinxhir, ose sic quhet ndryshe, një blockchain [5].

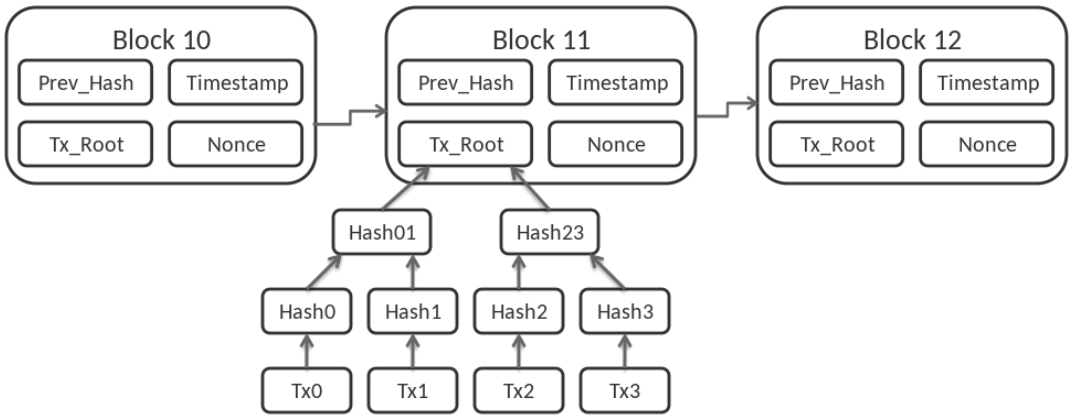


Figura 3.1 - Struktura e transaksionit të blockchain

Kryesisht, një blockchain menaxhohet nga një rrjet me shumë antarë, të cilët quhen "nodes", ku secili ruan një kopje të rekordeve. Kjo bën të mundur që në rast se një node kompromentohet, pjesa tjetër e nodeve dhe e blockchainit mbetet e pandryshuar dhe e mbrojtur ndaj sulmeve keqdashëse. Kjo për shkak se të gjitha nodet do të kenë kopje ndryshe nga ai node i kompromentuar. Idealisht një node do të lidhet me cdo node tjetër në sistem duke formuar një rrjet të shpërndarë (distributed). Por realisht një node ka një limit lidhjesh, duke e kthyer rrjetin në një sistem të decentralizuar. është provuar që një sistem i tillë është më se i mjaftueshëm për përdorime të niveleve të larta, sic janë menaxhimet e të dhënave sensitive ose transaksioneve dhe arkivave bankare.

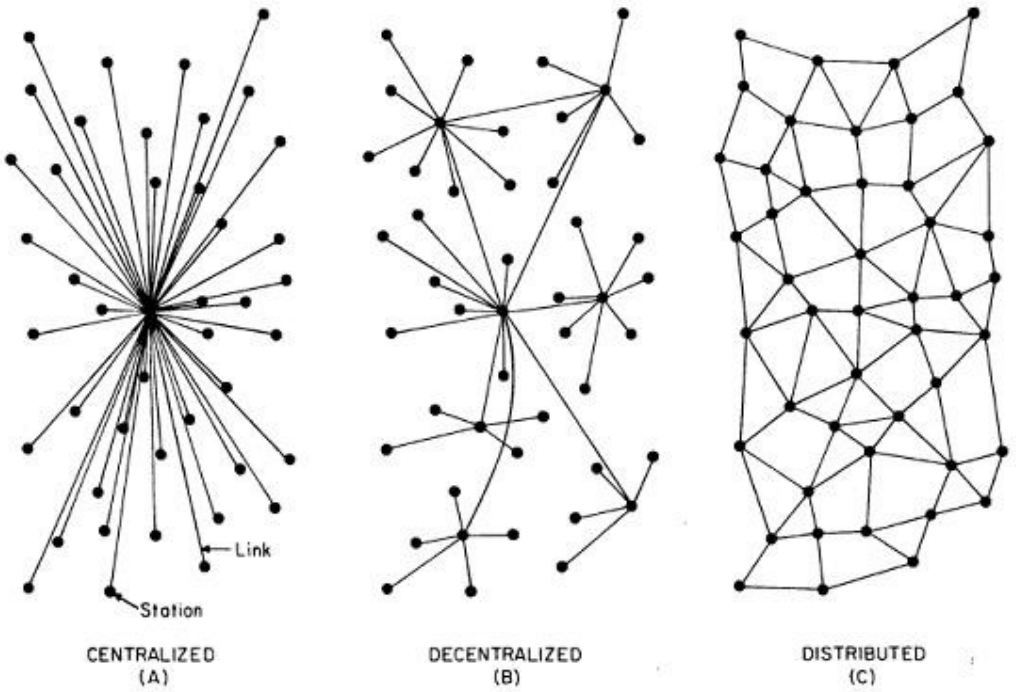
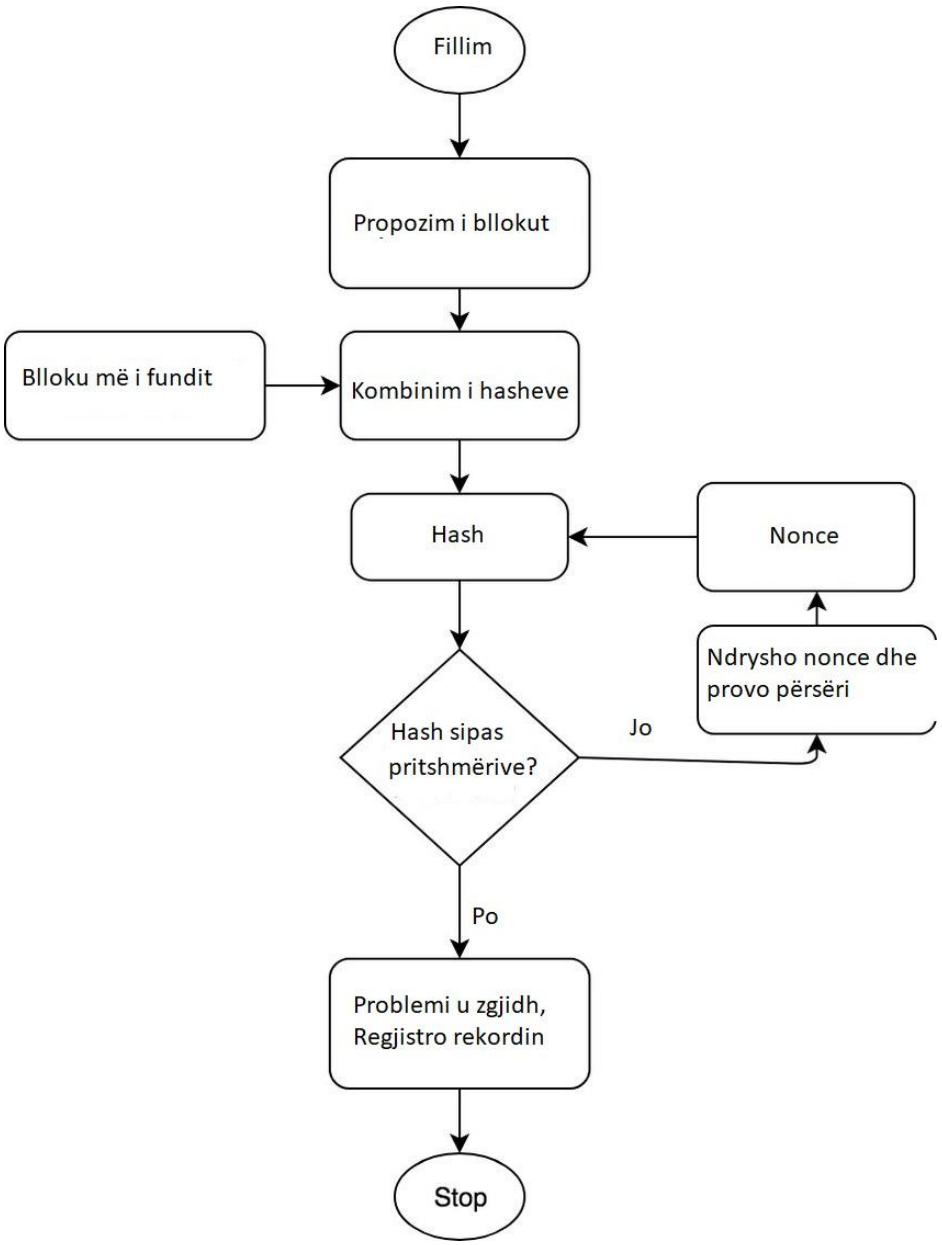


Figura 3.2 - Diferenca mes lidhjeve të ndryshme

Në figurën 3.2, rasti i parë përbën pjesën më të madhe të sistemeve aktuale, ku shihet qartë një pike e dobët. Në rast se ajo pikë bie, ose në rast se lidhja shkëputet, komunikimi ndërmjet pikave të tjera do të ishte i pamundur. Në rastin e një sistemi të decentralizuar kemi shumë pika të shpërndara në rrjet, dhe rënia e një pike ndikon në një përqindje shumë të vogël të sistëmit. Edhe pas ri-lidhjes së anëtarit me pjesën tjetër të sistemit, bëhet rivlerësimi i të dhënave dhe bëhet sinkronizimi me pjesën tjetër të sistemit [6].

3.1.1. PoW (Proof-of-Work)

Duke qënë se në sistem shtohen të dhëna vazhdimisht, duhet bërë një diferencim se cilat të dhëna duhet të shtohen dhe cilat jo, duke bërë kështu verifikimin e tyre. Në versionet më të hershme të blockchain përdoren metodoligji konsensusi sic është PoW (Proof-of-Work) ku një transaksion dhe bllok i ri në blockchain kërkon zgjidhjen e një problemi kriptografik nga antarët që janë në rrjet. Zgjidhja e parë që gjendet i komunikohet pjesës tjetër të rrjetit, kjo përgjigje verifikohet nga të gjithë antarët dhe më pas blloku konfirmohet dhe shtohet në rrjet [7].



3.4 - Skema e përfshirjes së bllokut në PoW

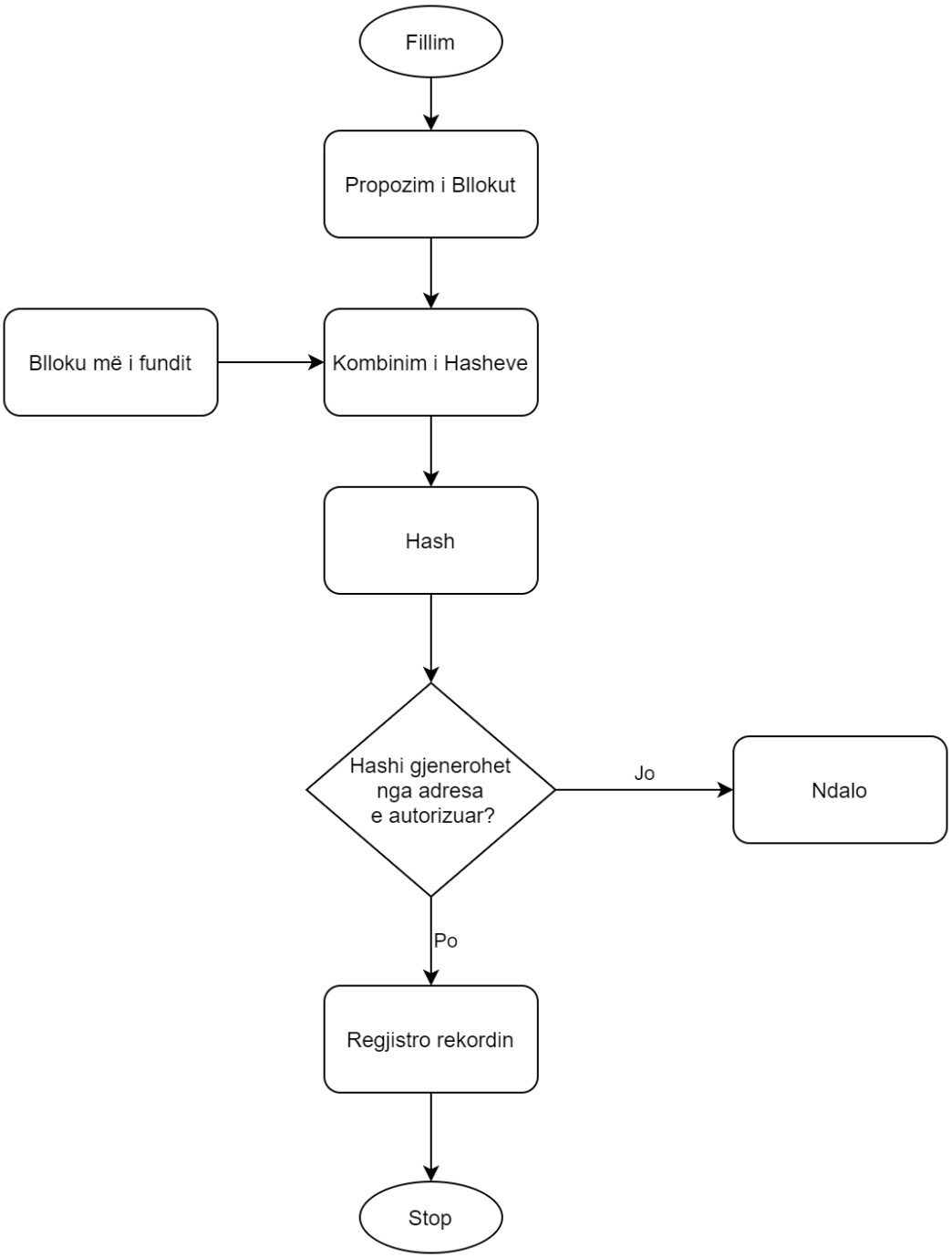
Nonce është një numër dhjetor i cili, së bashku me të dhënat e bllokut, ndihmon në krijimin e hashit (paraqitet si numër hexadecimal). Përgjithsisht ky nonce rritet me 1 deri sa hashi i marrë të kënaqi kërkesat e sistemit. Kushtet e hashit varen nga "vështirësia" e sistemit, e cila vendoset në konsensus me të gjithë anëtarët e rrjetit. Kryesisht kjo përcaktohet nga numri i zerove (0) me të cilat duhet të fillojë hashi i bllokut. Një shembull më se perfekt do të ishte nga rrjeti më i mall i blockchainit, nga rrjeti BITCOIN.

Blloku Nr.	Hash	Koha e Përfshirjes
9	000000008d9dc510f23…	2009-01-09 04:54
702151	0000000000000000000a18...	2021-09-25 15:52

Tabela 1.1 - Krahasimi mes dy blloqeve

Sic shihet nga tabela 1.1, në fillim kur vështirësia ishte e ulët kemi vetëm 8 zero në fillim të hashit. Kjo do të thotë se janë 16^8 mundësi për gjenerimin e hashin të bllokut Nr.9, dhe 16^19 për hashin e bllokut Nr.702151. Vështirësia e blockchain fillon me 1 dhe ndryshon cdo 2016 blloqe. Qëllimi është që 2016 blloqe të krijohen cdo 2 javë, dhe bazuar në këto parametra bëhet ndryshimi në vështirësi. Faktori i ndryshimit nuk kalon 4 ose ¼, kjo për të mos e bërë ndryshimin shumë drastik. PoW aktualisht është mekanizmi më i mirë i mbrojtjes së një blockchain publik, mbi të cilin janë vendosur shumë protokolle dhe implementime. Në rastin e PoW sistemi është tërësisht i pavarur, dhe kostoja për përfshirjen e transaksionit dhe informacionit në blockchain varion gjerësisht nga shumë faktorë globa, dhe i paguhet rrjetit të blockchainit.
Kryesisht në rastin tonë kjo do të thotë se lëshimi i certifikatave do të limitohet në kohë, madhësi dhe cmim. Pra në formën e tij më të thjeshtë, do mund të lëshonim vetëm një certifikatë cdo 3-4 minuta për një kosto të matshme. Duke marrë në konsideratë certifikatat në total, kjo është një zgjidhje e pamundur.

3.1.2. PoA (Proof-Of-Authority)

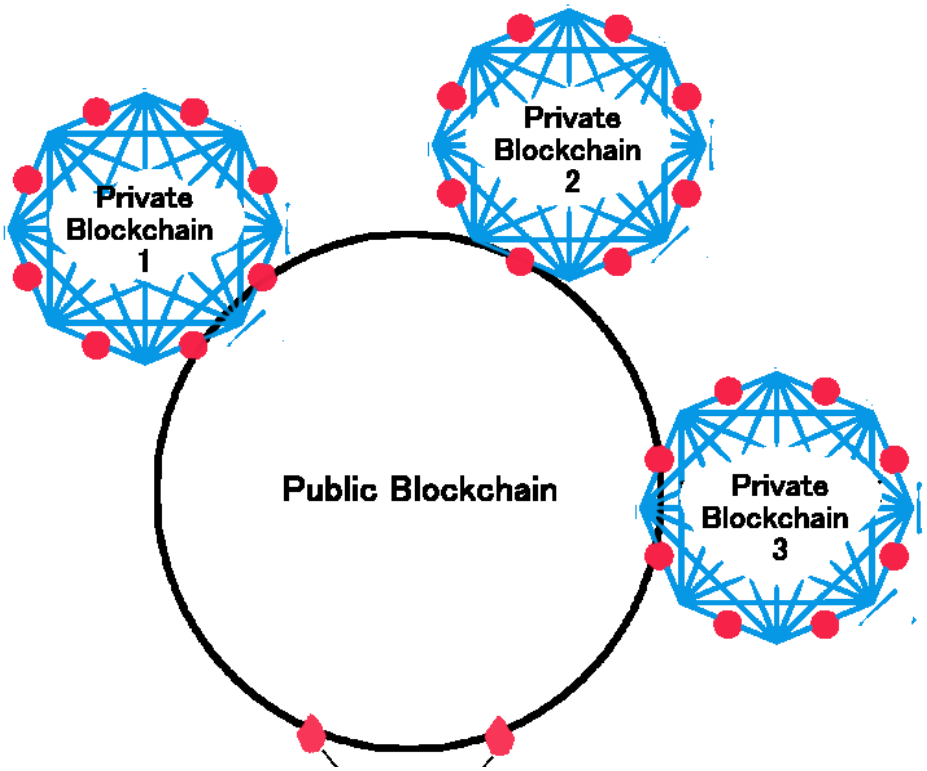


3.5 - Skema e përfshirjes së bllokut në PoA

Ky mekanizëm konsensusi lejon vetëm një antar të rrjetit të regjistrojë transaksione të reja i cili quhet "vërtetues", dhe pjesa tjetër e antarëve shërben për rritjen e integritetit të të dhënave. Kjo metodë është më e shpejtë, pasi nuk kërkohet zgjidhja e një problemi kriptografik, por limiton aksesin e anëtarëve të rinj të cilët mund të shtojnë të dhëna. Në rast se kërkohet të shtohet një vërtetues tjetër me adresë 0x… atëherë 50% e vërtetuesëve aktualisht në rrjet duhet të iniciojë veprimin për shtimin e vërtetuesit të ri [7]. Konfirmimi i blloqeve bëhet cdo x sekonda të cilat specifikohen në bllokun gjenezë të blockchainit. Zakonisht ky numër është cdo 15 sekonda, por mund të bëhet dhe cdo transaksion të ri. Pra kjo do të thotë se cdo bllok përfshin vetëm një transaksion.

Në rastin e PoA rrjeti varet nga një entitet ose grup entitetesh. Ndonëse do të ishtë shumë e vështirë për të bllokuar një rrjet blockchaini privat, nuk është e pamundur. Një shembull do të ishte falimentimi i kompanisë e cila merr përsipër menaxhimin e blockchainit që i përket entitetit në fjalë (në disa raste kjo kompani mund të jetë dhe entiteti i lëshimit të certifikatave). Këtu faktor kryesor është cmimi i blerjes së nodeve (antarëve). Kjo mund të ulet me lejimin kontributeve të jashtme, por gjithsesi mbetet në fokus. Ky konsensus për ne nuk përbën problem, pasi në të gjitha rastet lëshimet do të bëhen nga institucionet përkatëse.

3.1.3. Sistem Hibrid



3.6 - Kombinimi i llojeve të ndryshme blockchain

Mundësia e fundit do të ishte një implementim hibrid ose i kombinuar. Por kjo, për momentin do të shtonte kompleksitet dhe do të kishtë shumë pak benefite për përdorimin tonë. Një implementim i një sistemi të tillë do të përfshinte ruajtjen e pjesës më të madhe të informacionit në një blockchain privat, dhe pjesën më të rëndësishme dhe më kritike në blockchain publik. Shembull do të ishte një sistem arkivimi, ku dosjet (PDF, WORD, PPTX . . .) [8] ruhen në blockchain privat, ndërsa hashi i tyre dërgohet në blockchain publik, kështu duke krijuar një lidhje mes përmbajtjes së dokumentit dhe integritetit të saj.
Megjithëse kjo teknologji është fokusi primar, për zhvillimin dhe perdorimin e saj do përdoren disa teknologji të tjera suplementare.

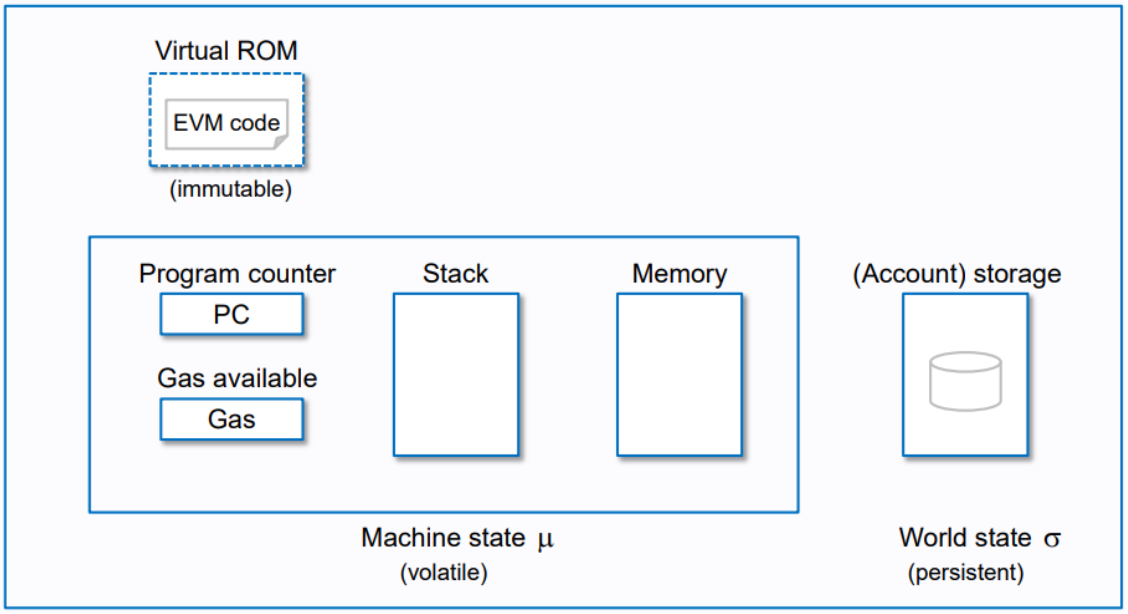
3.2. EVM

Solidity është një gjuhë programimi e orientuar nga objekti e krijuar gjatë vitit 2014, disi e ngjashme me Java. Me përdorimin e solidity futet një koncept tjetër, i cili është "smart contract" ose ndryshe kontrata virtuale. Këto nuk janë asgjë tjetër vecse kod i programuar për të kryer veprime të caktuara në bazë të veprimeve që kryhen nga përdoruesi. Këto kontrata veprojnë në sistemin e brëndshëm të EVM, dhe është pikërisht aty ku këto kontrata do të qëndrojnë pas lëshimit [9], dhe me këto kontrata do

të bëhet lëshimi dhe menaxhimi i certifikatave të pronësisë. Pavarësisht se blockchaini shërben si një databazë e shpërndarë, ne nuk kemi mundësinë e përdorimit të një strukture databaze të mirfilltë, kështu që duhet ta krijojmë një të tillë nëpërmjet kontratave virtuale, kjo në përputhshmëri me tipet e të dhënave që do të ruajmë. Jo vetëm kaq, por dhe mënyra e lëshimit si dhe kërkesat për verifikim do të menaxhohen nga kjo kontratë virtuale.

3.2.1. EVM (Ethereum Virtual Machine)

EVM është një vend përgjegjës për ekzekutimin e instruksioneve kompjuterike. Një analog i përshtatshëm do të ishte një server apo kompjuter personal, i cili ekzekuton instruktionet e marra nga përdoruesi dhe ruan të dhënat dhe outputin në disk. EVM është pikërisht një gjë e tillë, por nuk mund të imagjinohet kurrsesi si një entitet i vetëm. EMV përbëhet nga të gjithë antarët pjesëmarrës në blockchain, duke unifikuar operacionet e ekzekutimit të kodit dhe ruajtjes së të dhënave ndërmjet anëtarëve në një të vetëm.



3.7 - Struktura e brendshme e EVM

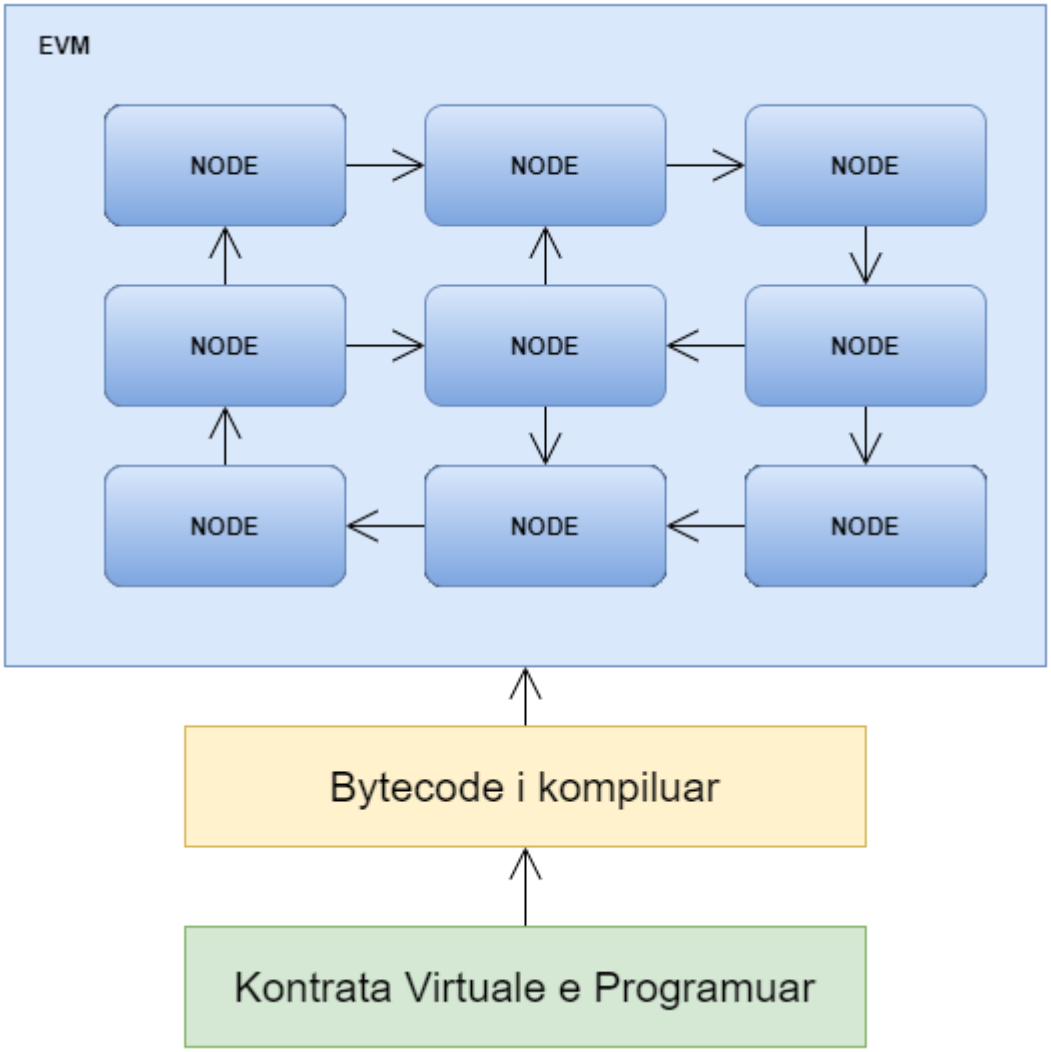
Sic shihet, struktura e EVM i ngjason një procesori të zakonshëm. Në rastin e EVM ekzekutimi i cdo instruktioni kushton dhe matet me “gas”. Rrjedhimisht cdo transaksion vjen me një specifikim i cili quhet gasLimit, i cili specifikon maksimumin e gas që merret nga dërguesi i transaksionit për ekzekutimin e tij.

Instruksioni	Cmimi
ADD	3 GAS
MUL	5 GAS
AND	3 GAS
OR	3 GAS
BALANCE	400 GAS

Tabela 3.1 - Cmimi i instruksioneve në gas

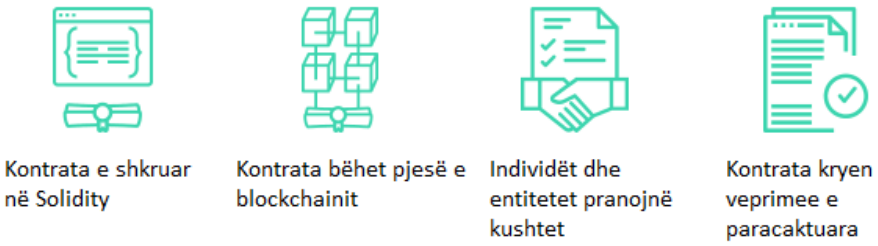
3.2.2. Solidity & Smart Contracts

Meqënëse EVM ndërtohet mbi një arkitekturë të nivelit të ulët (low-level), mund të krijohen gjuhë programimi të nivelit të lartë. Një e tillë është dhe solidity. Solidity është një gjuhë programimi e nivelit të lartë e orientuar nga objekti e krijuar gjatë 2014. Kjo gjuhë është e ngjashme me Java, C++, JavaScript dhe Python. Kryesisht përdoret për krijimin e kontratave smart dhe kompilohet në bytecode, i cili më pas ekzekutohet direkt në EVM.



3.8 - Kalimi nga kontratë në node

Në një nivel më të lartë, kontrata bën të mundur ekzekutimin e instruksioneve të parapërcaktuara nga programuesi pa pasur nevojë për ndërhyrje të mëvonshme. Me këto kontrata do të bëhet lëshimi dhe menaxhimi i certifikatave të pronësisë. Pavarësisht se blockchaini shërben si një databazë e shpërndarë, ne nuk kemi mundësinë e përdorimit të një strukture databaze të mirfilltë, kështu që duhet ta krijojmë një të tillë nëpërmjet kontratave virtuale, kjo në përputhshmëri me tipet e të dhënave që do të ruajmë. Jo vetëm kaq, por dhe mënyra e lëshimit si dhe kërkesat për verifikim do të menaxhohen nga kjo kontratë virtuale [9].



3.9 - Hedhja dhe përdorimi i kontratës virtuale

3.3. Teknologji Suplementare

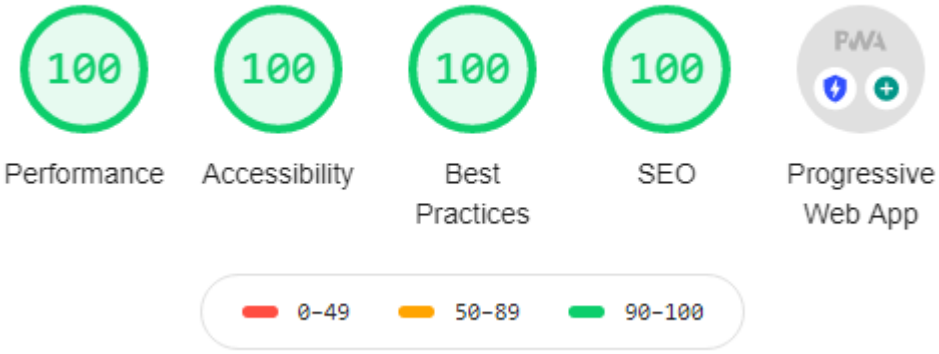
Megjithëse Blockchain në vetvete është një teknologji e fuqishme dhe me potencial të jashtëzakonshëm, ajo nuk mund të punojë e vetme. Për bashkimin e kësaj teknologjie me përdoruesin e zakonshëm do të nevojiten disa teknologji të tjera shtesë, të cilat do të bëjnë të mundur kalimin e informacionit nga përdoruesi tek blockchaini në një mënyrë të sigurt në përputhje me rregullat dhe standardet e sigurisë së informacionit digjital në një mënyrë sa më të thjeshtë për përdoruesin.

3.3.1. Web Apps & PWA

Për integrimin e blockchainit dhe kontratave virtuale në paisjet aktuale do të përdorim teknologji të vlefshme kudo. Të tilla janë programet e webit ose PWA (Progressive Web Apps), të cilat ndihmojnë në përdorimin më të thjeshtë të programeve vizuale, duke lehtësuar proceset e verifikimit dhe lëshimit. Programe të tilla do të komunikojnë indirekt me blockchain duke marrë të dhëna nga blockchain, dhe duke kryer veprime sipas kushteve të vendosura në kontrata virtuale. Këto programe do të shërbejnë si hyrje kryesore e përdoruesve të projektit (punë diplomës). Një veprim i tillë mund të jetë verifikimi i një certifikate pronësie, skanimi i saj apo lëshimi nga entiteti përkatës.
Kryesisht përdoren aplikacione specifike (native) për një platformë, por kjo kërkon shumë punë, pasi duhet ndërtuar një program specifik për cdo platformë (Linux, Windows, Android, iOS), por në këmbim kemi performancë të lartë. Ky rast nuk është eficient për ne.
Nga ana tjetër kemi faqet web, të cilat janë të lehta për tu ndërtuar, hapen njëlloj në të gjitha platformat, por nuk kanë opsionet që na duhen dhe nuk mund të instalohen nëpër paisje.
Këtu mund të përdorim PWA, një teknologji e viteve të fundit (Në fillim të 2021 PWA morën suport dhe nga iOS, e cila ishte e vetmja platformë që nuk e njihte zyrtarisht si teknologji), ku ofrojnë pjesën më të madhë të opsioneve specifike të plaformave, janë të qëndrueshme dhe mund të instalohen. Në fakt PWA nuk është një teknologji e vetme, por një kombinim i disa teknologjive të tjera, dhe i ngjan më shumë një filozofie për ndërtimin e aplikacioneve të webit [10].

3.10 - Teknologjitë e ndryshme ndaj grafikut kohë/mundësi

Për krijimin e PWA duhet të plotësohen disa kushte të cilat janë më se të përmbledhura në një mjet auditimi sic është Google Lighthouse.

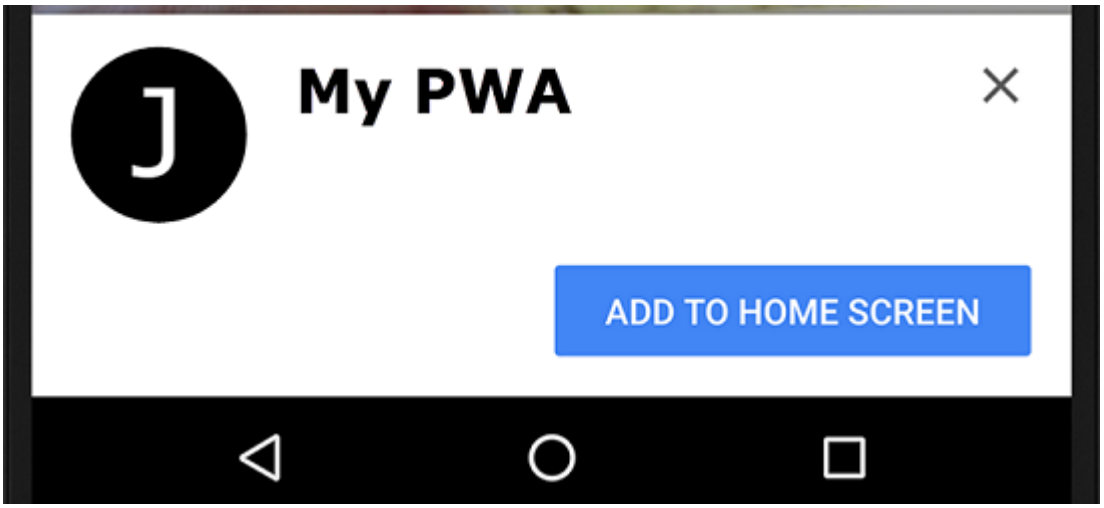


3.11 - Shembull i një PWA me vlerësim 100%

Fotoja më sipër i përket një PWA e cila është në përputhshmëri me kushtet e PWA (Gusht 2019). Kushtet e listuara nga Lighthouse janë si më poshtë:

- Performanca
  - First Contentful Paint (Koha kur pixeli i parë shfaqet në ekran)
  - Speed Index (Përqindja e mbushjes vizuale të faqes përgjatë kohës)
  - Largest Contentful Paint (Koha për shfaqen e tekstit ose fotos më të madhe)
  - Time to Interactive (Koha kur aplikacioni është gati për ndërveprime)
  - Total Blocking Time (Totali i kohëve ndërmjet FCP dhe TTI, vetëm mbi 50ms)
  - Cumulative Layout Shift (Zhvendosja e elementeve vizuale në ekran)

- Aksesibiliteti
  - Titull
  - Butona të aksesueshëm
  - Atributi lang në html.
  - Headers të sortuar
  - Praktika të tjera të rekomanduara
- Praktika të mira
  - HTTPS
  - Librari me probleme sigurie
  - Mbrojtje ndaj XSS
  - Errore në console
  - Praktika të tjera të rekomanduara
- Progressive Web App
  - E instalueshme
  - Përfshin file manifest
  - Regjistron service worker
  - Redirekton HTTP në HTTPS
  - Pritje në ekranin e ngarkimit



3.12 - Shembull i një butoni instalimi për PWA

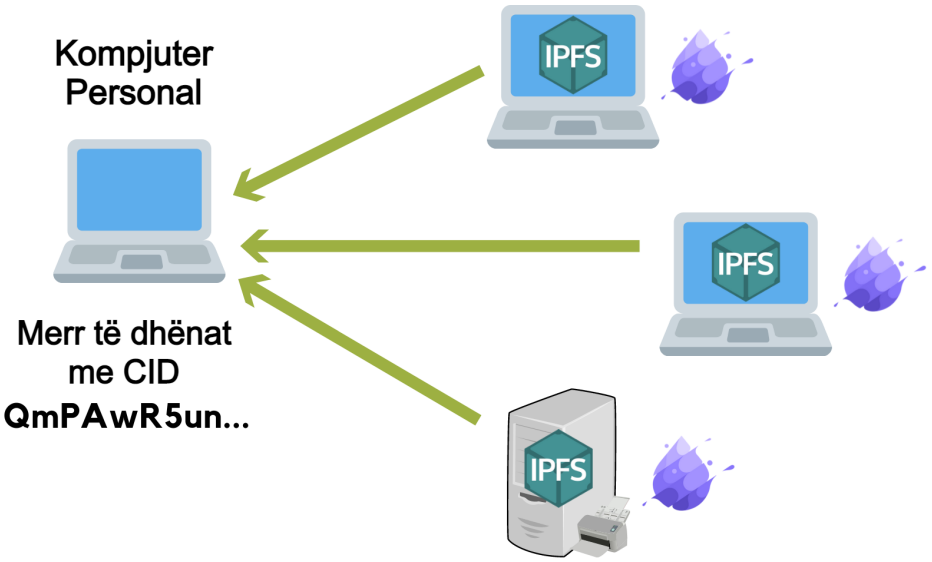
Pasi plotësohen të gjitha kushtet, në ekranin e përdoruesit shfaqet mundësia për ta instaluar aplikacionin

3.3.2. IPFS (InterPlanetary File System)

IPFS është një mënyrë për të përfshirë akoma më shumë të dhëna në një certifikatë. Bazohet mbi një protokoll peer-to-peer për ruajtjen dhe marrjen e të dhënave në sistem të shpërndarë. IPFS përdor adresim përmbajtje për të identifikuar cdo file. E ngjashme me BitTorrent, të dhënat janë shpërndarë ndërmjet shumë anëtarëve të rrjetit, dhe mund të merren duke përdorur DHT (distributed hash table) [11]. IPFS bazohet mbi këto parime:

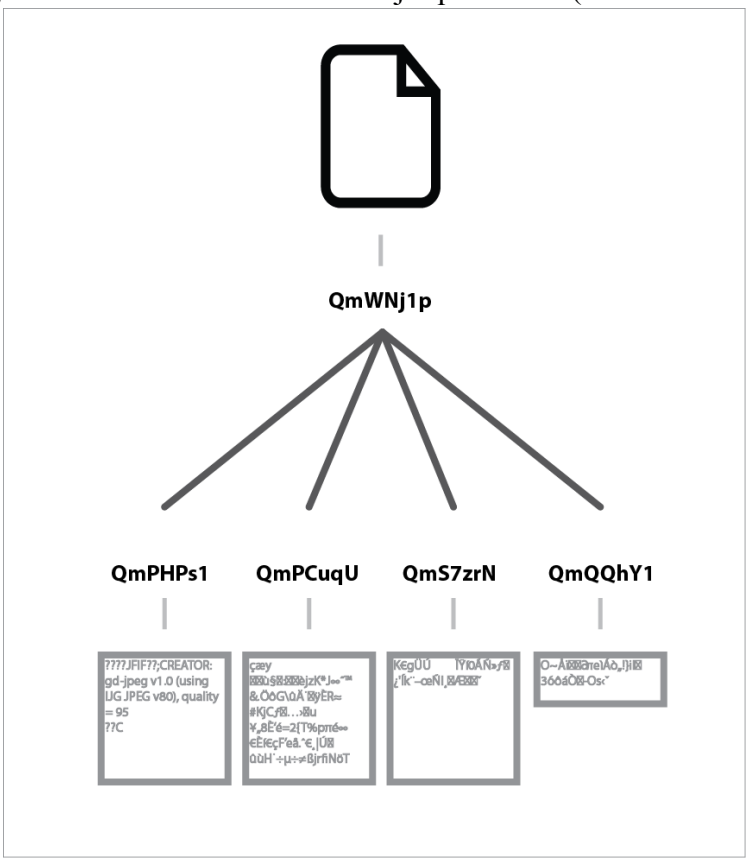
- Identifikim bazuar në përmbajtje
- Lidhje nëpërmjet Directed Acyclig Graphs (DAG)
- Zbulim dhe gjetje të dhënash me Distributed Hash Tables (DHT)

Content addressing përdoret për të identifikuar të dhënat sipar përmbajtjes dhe jo sipas vendndodhjes. Normalisht në webin e sotëm përmbajtja gjendet sipas kësaj të fundit. Le të themi se duam propozimin e Bitcoin. Kjo do të ishte dicka e ngjashme si më poshtë: https://bitcoin.org/bitcoin.pdf



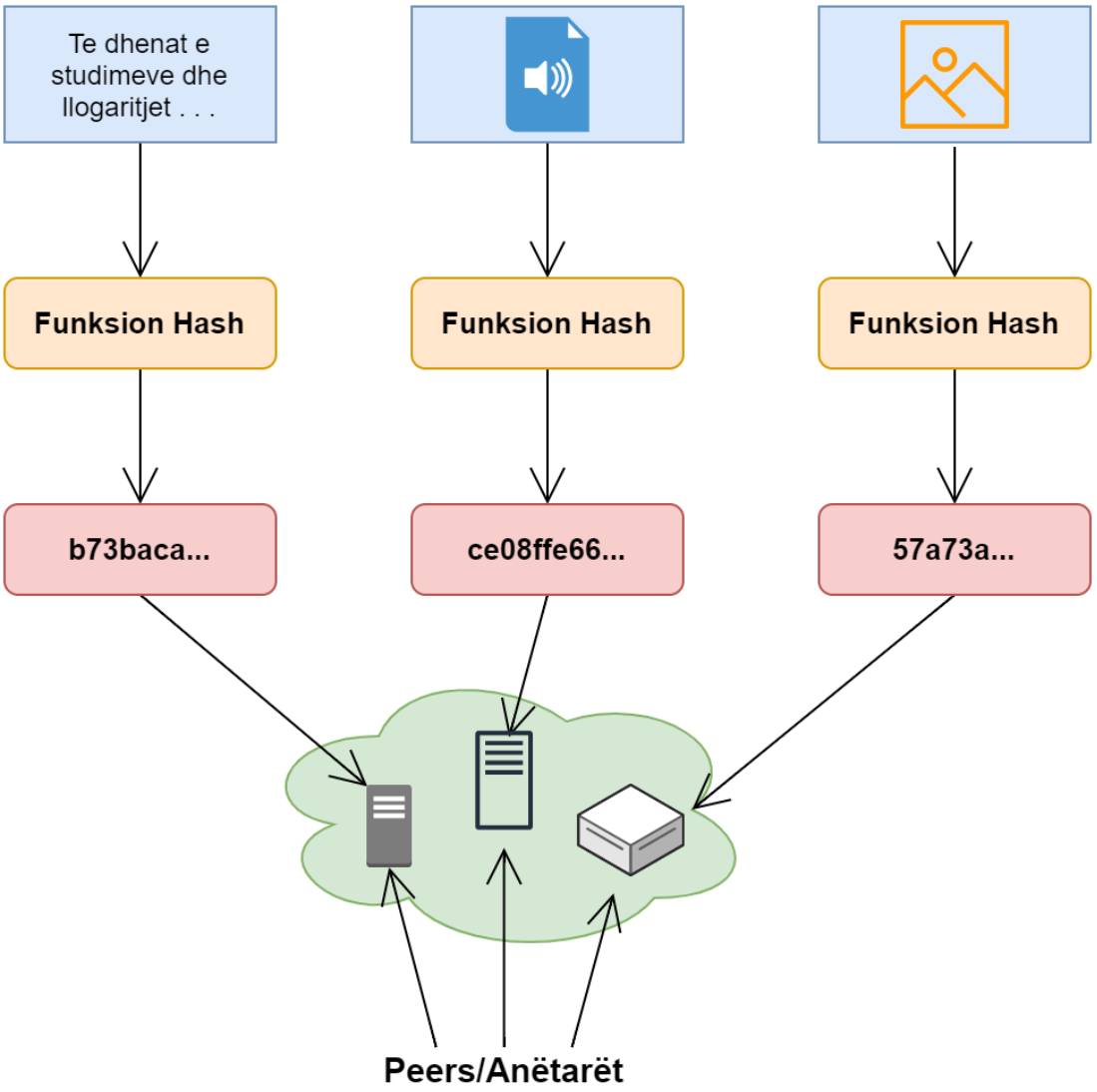
3.13 - DHT Skematikisht në IPFS

Për të aksesuar këtë pjesë informacioni na duhet vendndodhja. IPFS funksionon ndryshe, duke i vendosur një titull kësaj përmbajtje. Ky titull është një hash i gjeneruar sipas përmbajtjes së informacionit. Në IPFS kjo quhet CID (Content Identifier).



3.14 - DAG Skematikisht në IPFS

Më pas përdoren Directed Acyclic Graphs për lidhjen e informacionit me njëri tjetrin. Kjo kryesisht për të përfaqësuar lidhjen mes direktorive dhe dokumenteve.

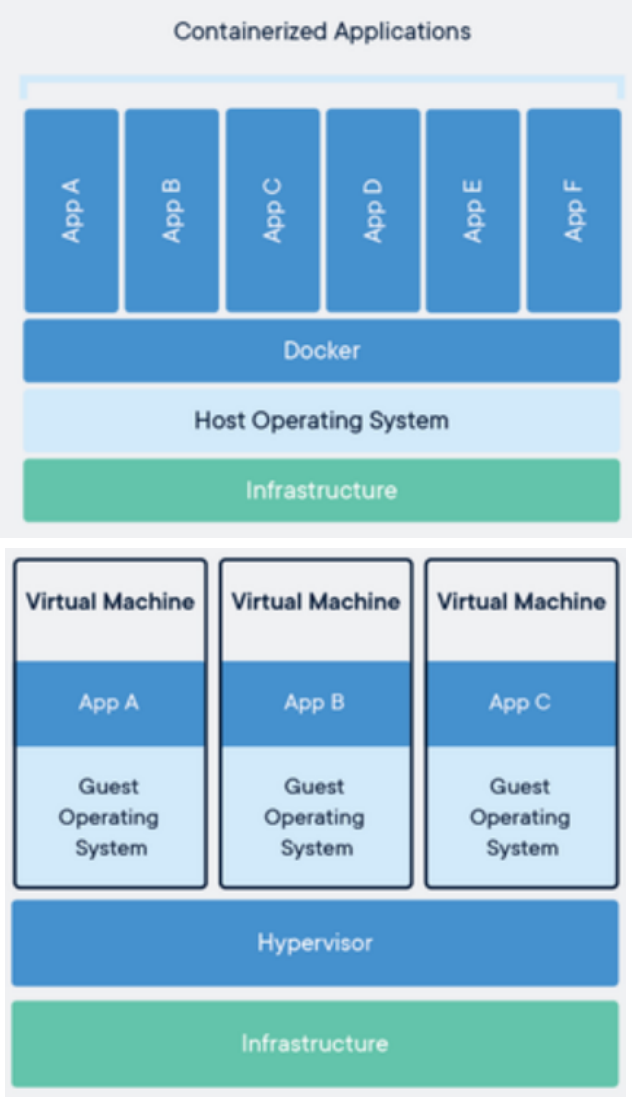


3.15 - Kalimi nga dokument në IPFS

Së fundmi përdoret Distributed Hash Tables për të gjetur se cili anëtar ruan informacionin që po kërkohet.

3.3.3. Docker Containers

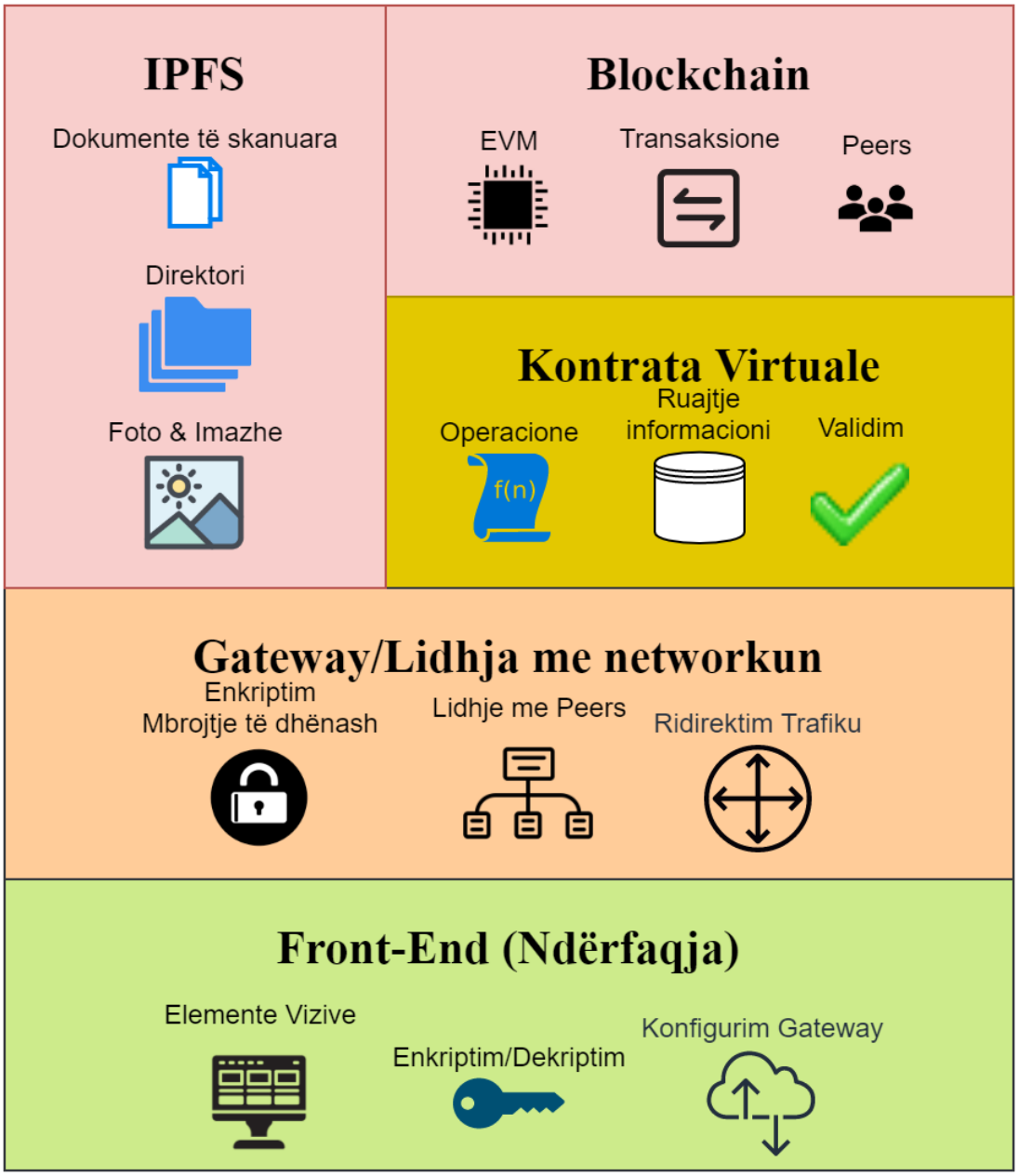
Docker është një platformë për zhvillimin, hedhjen dhe ekzekutimin e aplkiacioneve në një mjedis të kontrolluar dhe me mirëmbajtje minimale. Docker synon të eleminojë variablat shtesë, sic janë versionimi i sistemit të operimit, versionimi i paketave, vetë sistemi i operimit etj. Cdo program i hedhur në docker duhet të sillet njëlloj nëpërmjet cdo sistemi operimi [12]. Ne do ta përdorim këtë teknologji për të bërë hedhjen e nodeve në rrjet pothuajse të menjëhershme, duke minimizuar konfigurimet fillestare të nevojshme për nodet.



3.16 - Diferenca mes Docker dhe Virtual Machine

Ndryshe nga një makinë virtuale (VM), e cila ekzekuton një sistem operimi të tërë, docker virtualizon vetëm në nivel softueri. Kjo ul ndjeshëm përdorimin e resurseve të serverit apo mjedisit ku do të jetë një node. Për më tepër, duke përdorur mjete si docker-compose, mund të krijojmë një mjedis të tërë duke përfshirë jo vetëm një node, por edhe një databazë, webserver apo gateway.

KAPITULLI 4: ARKITEKTURA



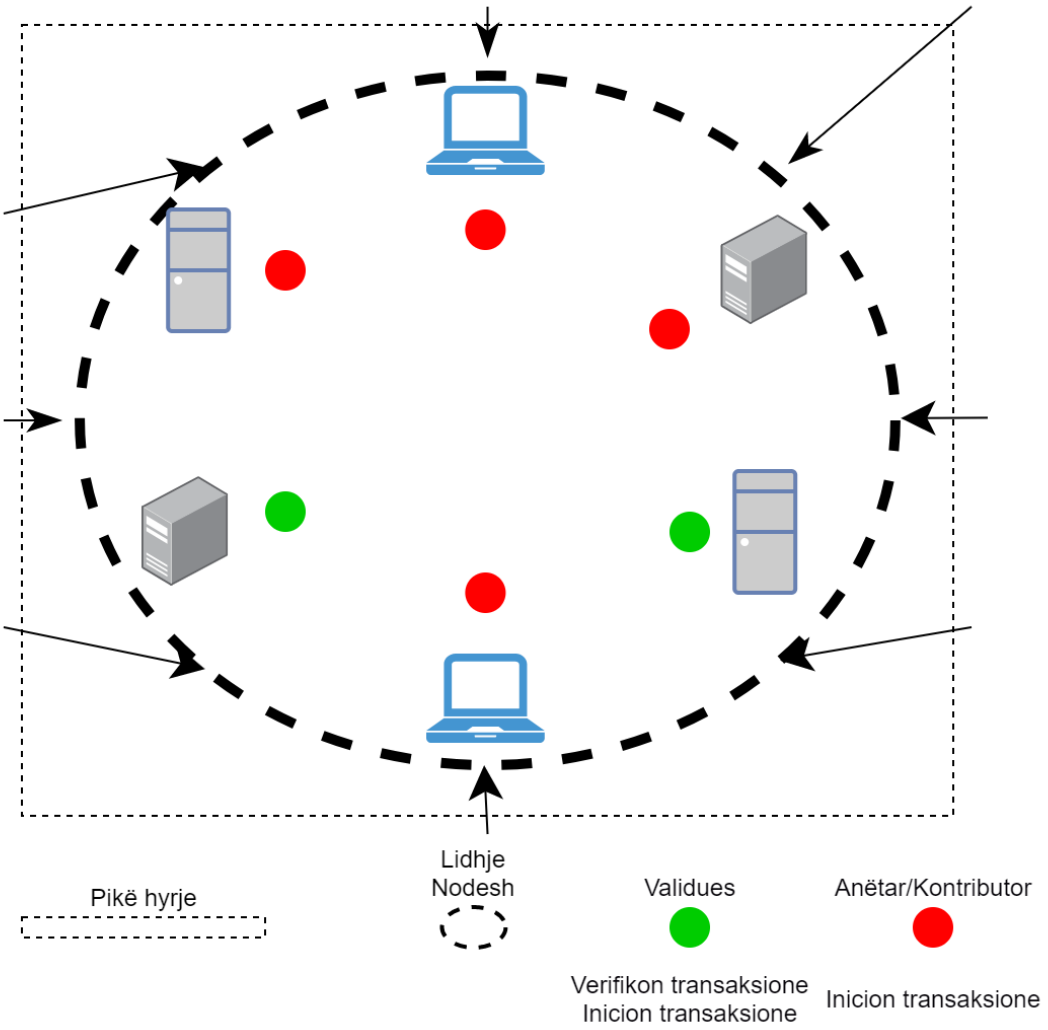
4.1 - Arkitektura e përgjithsme e projektit

Duke supozuar se projekti është përfunduar dhe është hedhur në produksion, entry-point ose pika e hyrjes së informacionit do të jetë në front-end. Ky informacion do të manipulohet dhe do të dërgohet në gatewayt përkatës. Në bazë të llojit të informacionit, i cili është i klasifikuar paraprakisht, ai do të dërgohet në networket përkatës, IPFS ose Blockchain. Duhet pasur parasysh se gjatë gjithë kësaj kohe informacioni përvesce enkriptohet nga përdoruesi përpara dërgimit, ai gjithashtu enkriptohet edhe me HTTPS gjatë rrugëtimit të tij nëpër internet. Rëndësi merr fakti se asnjë celës enkriptimi NUK DËRGOHET NE RRJET. Kjo rrit ndjeshëm sigurinë e të dhënave, por gjithashtu rrit mundësinë e këtyre të dhënave për tu bërë të papërdorshme në rast se ky celës humb. Pasi dërgohet në gatewayt përkatës, ky informacion ndahet dhe secili gateway dërgon informacionin në anëtarët përkatës të cdo networku apo rrjeti. Në rastin e IPFS ai ruhet direkt në rrjet, ndërsa në rastin e blockchain ai kalon nëpër metodat paraprake të

kontratës virtuale. Kujdes maksimal duhet pasur në këtë rast, sepse cdo informacion që hyn në kontratë është publikisht i shikueshëm dhe i adresueshëm. Për këtë arsye merren masa paraprake për mosidentifikimin e informacionit, sic janë enkriptimi i informacionit dhe përdorimi i adresave me jetëgjatësi të shkurtër. Në rastin tonë këto adresa nuk nevojiten.

4.1. Rrjeti Blockchain

Në rast se do zgjedhim një blockchain publik, nuk do të na duhet të bëjmë asnjë konfigurim për rrjetin, por le të themi se infrastrukturën duam ta bazojmë në një blockchain privat. Kjo do të rrisë disi punën që do duhet të bëjmë.



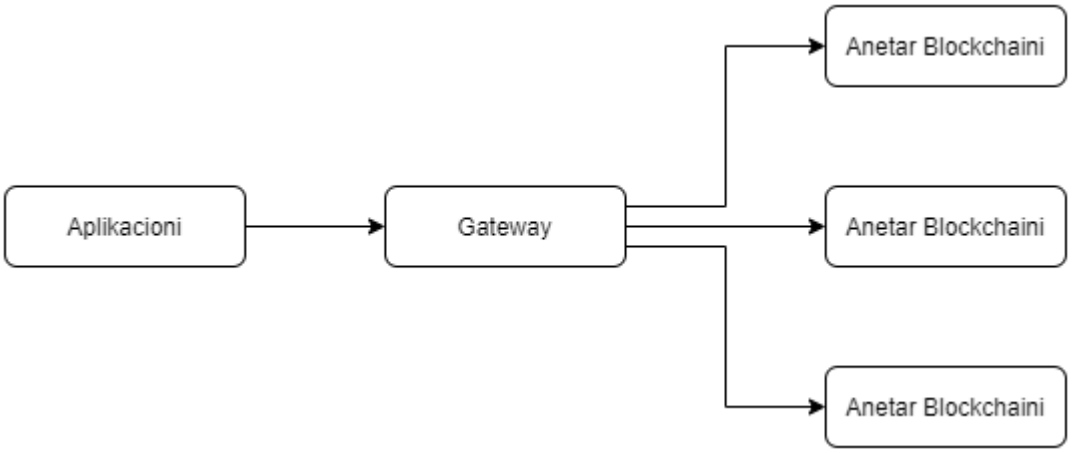
4.2 - Lidhja e përgjithshme mes anëtarëve

Një konfigurim si më lart do të përdoret, ku vetëm disa anëtar të selektuar do të kenë akses të konfirmojnë dhe të përfshijnë transaksione të reja në bllok. Këto anëtar specifikë do të quhen validues. Këta validues përvecse bëjnë konfirmimin e blloqeve të reja, ata gjithashtu ruajnë të dhënat e blockchainit dhe kryejnë operacionet e marra nga kontrata në EVM. Anëtarët që nuk luajnë rol në verifikimin e transaksioneve, por vetëm ruajnë të dhëna dhe shtojnë integritetin e sistemit do ti quajmë kontributorë [13].
Blockchain privat do të bazohet në Proof-of-authority me konsensusin `Clique`, ku verifikimet dhe shtimet në blockchain kryen vetëm nga disa adresa specifike. Në këtë

rast nuk do të kemi kosto në Lek (apo USD) për kryetjen e transaksioneve, si dhe do të bllokohet komplet ndërveprimi i paautorizuar me blockchainin. Kjo pasi adresat që bëjnë verifikimin e transaksionit janë në pronësi të sistemit tonë. Me Clique do të kemi një mënyrë më të lehtë për minimin e blloqeve, dhe do të ulet ndjeshëm nevoja për energji kompjutacionale. Në rast shtimi të adresave të reja, më shumë se ½ e adresave aktuale në rrjet duhet të jenë dakort për shtimin e adresës së re.

4.1.1 Gateway/Porta hyrjeje

Si shtesë do na duhet edhe një gateway, ose ndryshe portë komunikimi. Kjo pasi komunikimi i paisjes (telefon, kompjuter) me protokollin e blockchain nuk është i mundur në mënyrë direkte. Gateway është një portë që lidh aplikacionin tonë me rrjetin e blockchain, duke shërbyer si një urë komunikuese [14].
Ndërkohë cdo anëtar mund të shërbejë si gateway ose pikë hyrje në rrjet. Kjo bëhet duke ekspozuar një API që shërben si pikë hyrje, e cila fut informacionin në lidhjen e nodeve.

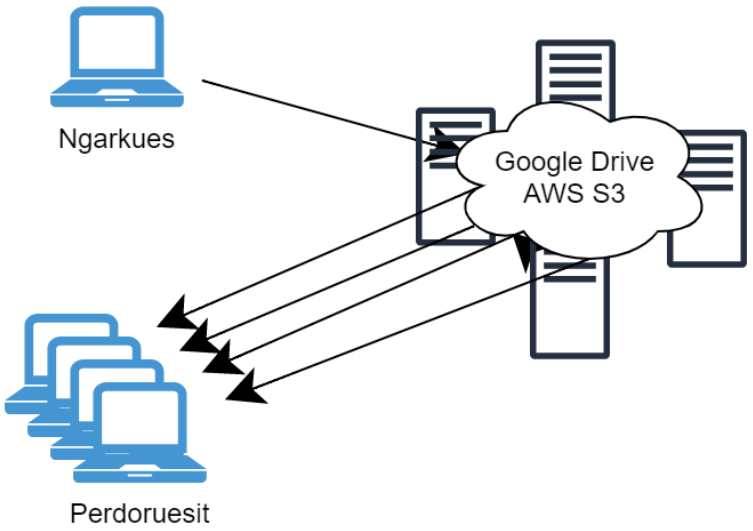


4.3 - Lidhja aplikacion-blockchain

Zgjedhja e një gateway është në dorën e përdoruesit, ekzistojnë alternativa të tilla si CloudFlare apo Infura të cilat ofrohen pa kosto ekstra, por për rastin tonë do të krijohet një gateway privat në funksion të blockchainit privat. Përdoruesi do të ketë një nga gatewayt e mësipërme, por duhet të dijë se është i lirë të ndryshojë atë gjatë gjithë kohës, duke lejuar për transparencë maksimale. E njejta gjë vlen edhe për IPFS.

4.2. Rrjeti IPFS

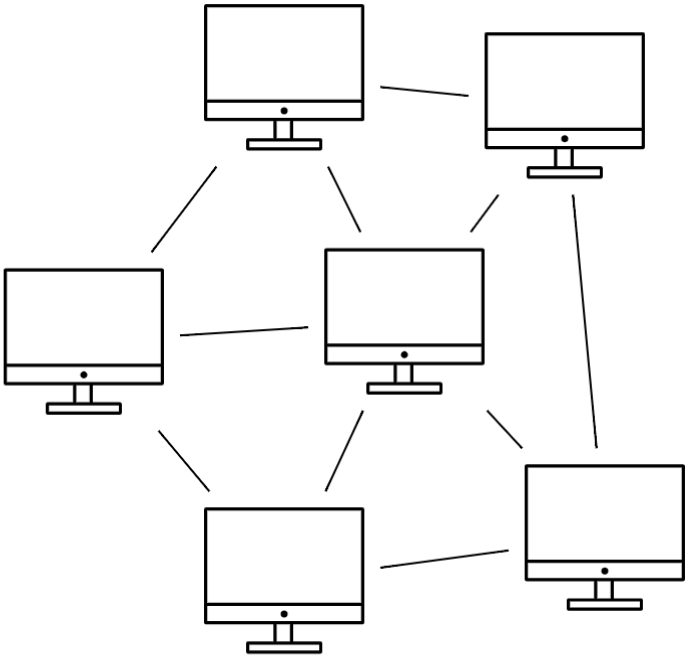
Rrjeti IPFS është i ngjashëm me atë të blockchainit. Por këtu mungon diferencimi mes validuesve dhe kontributorëve. Në këtë rast cdo anëtar mund të marrë dhe japë të dhëna. Mjafton që kërkesa për marrjen e të dhënave të përfshijë hashin e tyre. Për të mënjanuar mbingarkesën e sistemit përdoren mjete të avancuara sic janë orkestrimet, persistenca, apo ngulitja (pinning) [11][15].



Sisteme Tradicionale

4.4 - Ruajtja e informacionit në sisteme tradicionale

Sistemet tradicionale zakonisht mirëmbahen nga një organizatë e vetme. Dhe këtu përdoruesi komunikon direkt me një server, i cili ruan të dhënat dhe i shpërndan ato tek përdoruesit e tjerë



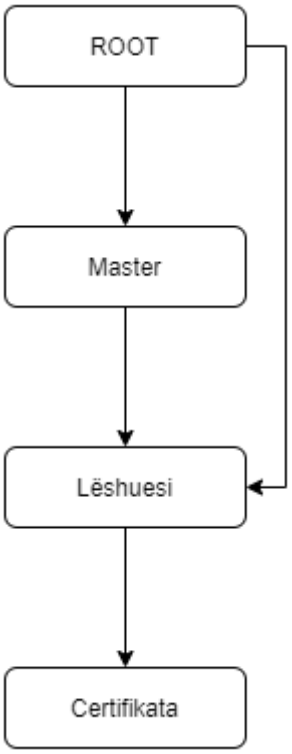
IPFS

4.5 - Ruajtja e informacionit në IPFS

Ky nuk është rasti i IPFS. Sic u tha më parë, të dhënat ruhen ndërmjet disa anëtarëve. Në rastin tonë ky do të jetë konfigurimi i anëtarëve të IPFS. Në këtë rast cdo anëtar shërben gjithashtu si pikë hyrje ose gateway.

4.3. Kontrata Virtuale

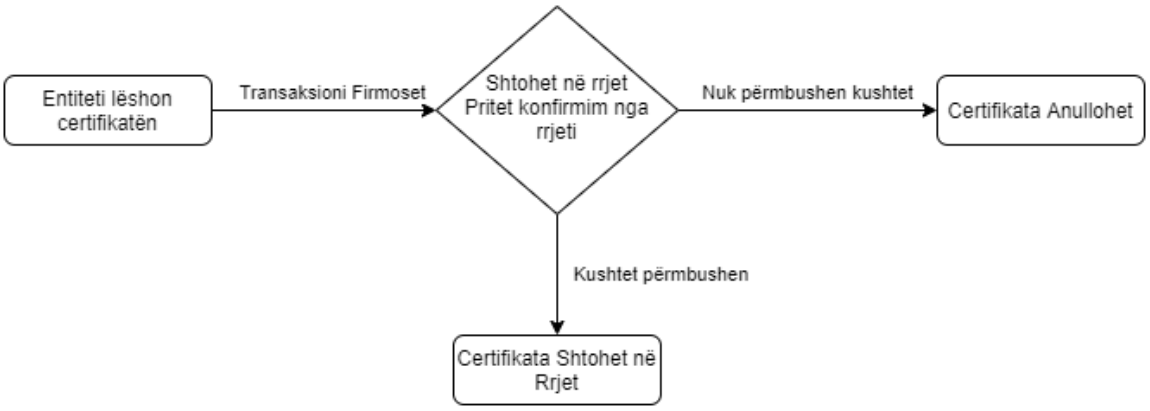
4.3.1. Ndarja e Roleve



4.6 - Konfigurimi i roleve

Zgjidhja e propozuar konsiston në krijimin e roleve. Pra do të krijohet një sistem i bazuar në role, ku identifikuesi kryesor do të jetë adresa e blockchainit.
Në lëshimin e kontratës virtuale duhet të plotësohet një parametër të cilin e quajmë "adresa master". Por cfarë është kjo adresë dhe përse do përdoret? Kjo adresë është adresa që do të bëjë verifikimin dhe lëshimin e entiteteve të tjera certifikuese. Pra kjo adresë do të jetë në pronësi të entitetit më të lartë përgjegjës të sistemit.
Një tjetër adresë kryesore do të jetë adresa ROOT. Kjo do të jetë adresa që do të bëjë lëshimin e kontratës virtuale. Pas lëshimit të kontratës virtuale kërkohet që celësi privat (private key) i kësaj adrese do të enkriptohet, ndahet në pjesë dhe do të ruhet në mënyrë sa më të sigurt. Arsyeja përse duhet një siguri kaq e madhe është sepse kjo adresë do të shërbejë si "siguresë" në rast se humbet ose kompromentohet adresa "master". Duke qënë adresa me nivel më të lartë hierarkik, sygjerohet që të ruhet offline e kriptuar jasht aksesit të lehtë dhe publik. Teknikat kryesore të ruajtjes përfshijnë ruajtjen në hapësirën lokale të serverit privat, offline në USB, disk CD/DVD, Letër ose duke përdorur shërbime eknas për ruajtjen e të dhëave sensitive.
Adresa tjetër do të jetë adresa e entitetit lëshues, e cila do të përdoret më së shumti. Kjo pasi ajo do të jetë përgjegjëse për lëshimin dhe verifikimin e certifikatave. Për lehtësi përdorimi kjo adresë do të ketë identifikues si emri dhe id e entitetit.

4.3.2. Verifikimi i certifikatave



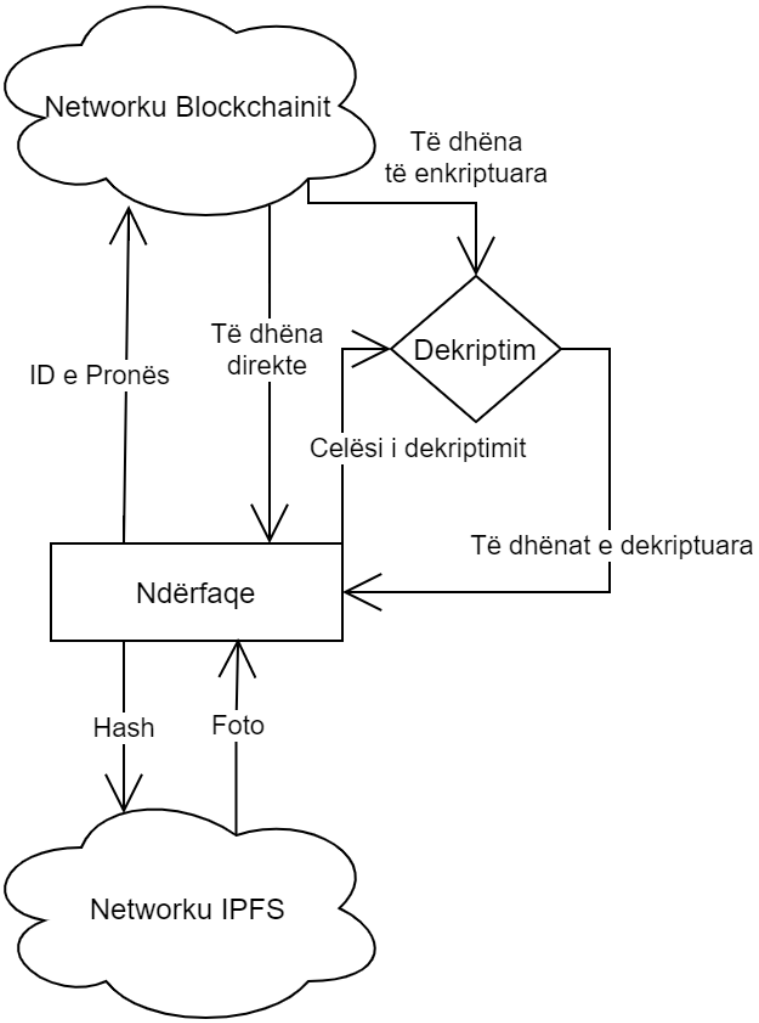
4.7 - Procesi i përdorimit të certifikatës

Në foton 2.4 shohim skematikisht mënyrën e shtimit të certifikatës në blockchain. Fillimisht entiteti lëshues lëshon certifikatën nga adresa e tij duke e firmosur atë me celësin e tij privat, dhe më pas kjo certifikatë shtohet në rrjet dhe pritet për verifikimin dhe shtimin e saj në një bllok. Në rast se pranohet nga rrjeti, më pas kryhet kontrolli i dytë ndaj kontratës virtuale, ku shikohet nëse certifikata është sipas kërkesave dhe nëse lëshuesi është i autorizuar. Nëse cdo gjë është në rregull atëherë kryhen modifikimet e nevojshme sipas instruksioneve të kontratës virtuale, dhe certifikata shtohet në rrjet. Në rast të kundërt transaksioni i lëshimit të certifikatës mbetet i konfirmuar, por procesi ndalon para shtimit të certifikatës. Pra kjo na jep anullimin e certifikatës dhe nuk konsiderohet nga kontrata virtuale, që do të thotë se praktikisht certifikata nuk është lëshuar me sukses. Në të dyja rastet merret e njëjta taksë nga sistëmi, kështu që duhet treguar kujdes nga cila adresë kryhet lëshimi.
Për shkak të kostos dhe limiteve në madhësi që kemi gjatë lëshimit të transaksionit do na duhet që të minimizojmë të dhënat që hidhen në blockchain. Kjo do të bëhet duke zhvendosur të dhënat si psh foto, video apo dokumente të tjera në rrjet tjetër. Qoftë kjo në IPFS ose server privat, ku në të dyja rastet do të ruhet hashi (vlerë unike për cdo skedar/dokument) në blockchain, duke bërë të mundur verifikimin e integritetit të dokumentit në shqyrtim.
Duhet theksuar sërisht se në cdo rast të dhënat do jenë të enkriptuara dhe do të jetë e pamundur për të parë të dhënat e certifikatës në rrjet. Vetëm të dhënat e transaksionit dhe blloku ku qëndron certifikata do të jenë publike. Kjo për të ruajtur privatësinë e përdoruesve.

4.4. Ndërfaqet

Këto ndërfaqe do të shërbejnë për komunikimin me rrjetin e blockchain. Ato duhet të jenë sa më të thjeshta në përdorim, pasi qëllimi është që të përdoren nga një publik i gjerë.

4.4.1. Verifikuesi

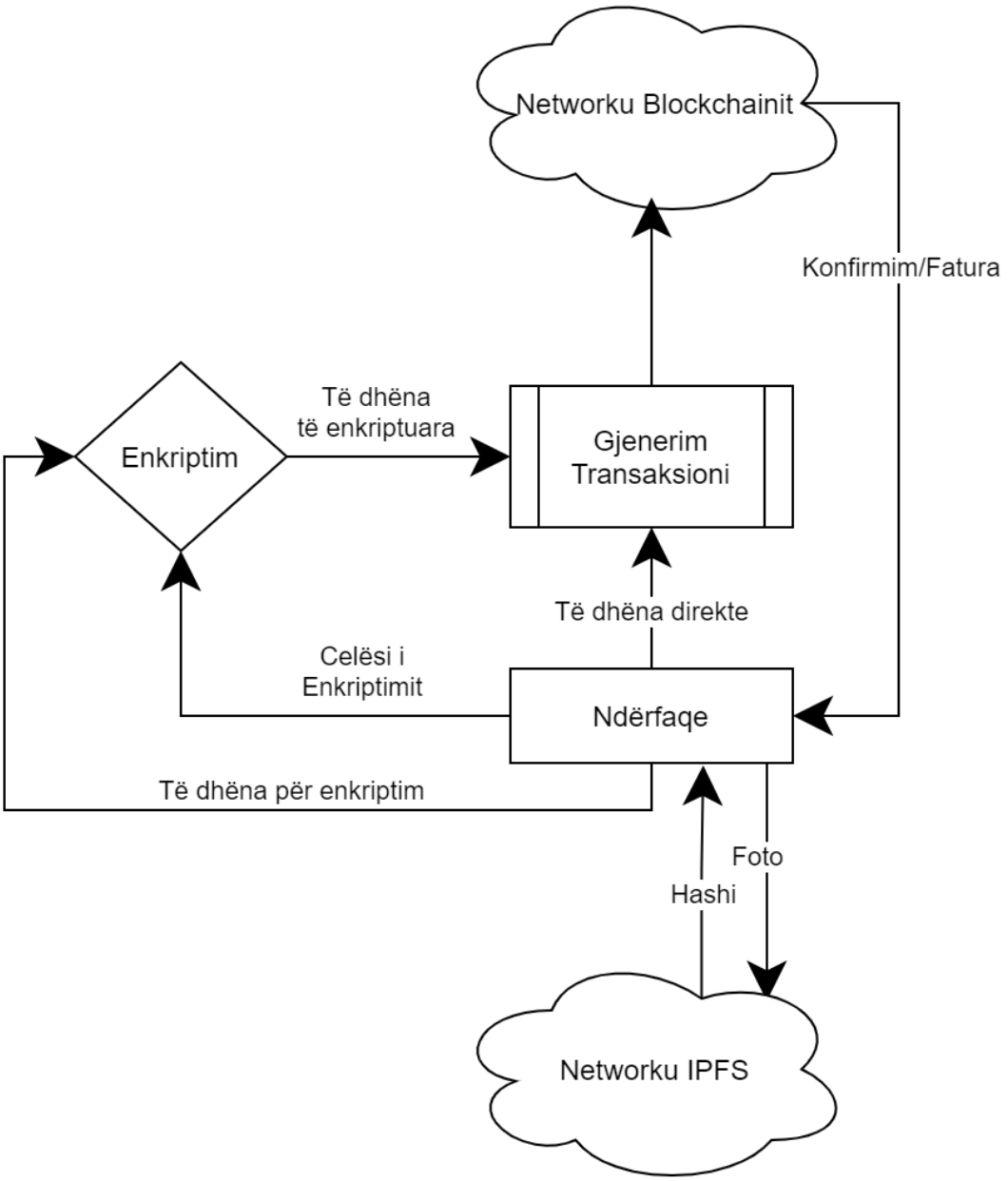


4.8 - Procesi i kalimit të informacionit nga verifikuesi në rrjetet përkatëse

Verifikuesi do të bëjë të mundur verifikimin e një certifikate. Si input do të merret ID e certifikatës së pronësisë. Në fillim do të shfaqen të dhënat publike, sic mund të jenë lloji i pasurisë, sipërfaqja ose përshkrim i përgjithshëm, adresa ose vendndodhja e përgjithshme e pasurisë. Në rast se pala e tretë mjaftohet me këto të dhëna, nuk procedohet më tej. Në rast të kundërt përdoruesi bën dekriptimin e certifikatës me celësin përkatës. Në këtë rast shfaqen edhe të dhënat e tjera private sic mund të jeni: Pronari, Bashkëpronarët, Zona kadastrale / Vendndodhja e përpiktë, Vlera e pasurisë etj. Duhet theksuar se këto të dhëna do të jenë në përputhje me ligjet aktuale, dhe se privatësia e tyre do të jetë lehtësisht e ndryshueshme. Në këtë rast lidhja me networkun e IPFS bëhet fill pasi merret hashi i fotos nga blockchaini.

4.4.2. Lëshuesi i certifikatave

Kjo pjesë e softuerit do i shërbejë institurionit lëshues. Në këtë ndërfaqe do të lëshohen certifikatat dhe do të ruhen në blockchain. Direkt pas lëshimit institucioni është përgjegjës për komunikimin e celsit të dekriptimit tek përdoruesi/pronari.



4.9 - Procesi i kalimit të informacionit nga verifikuesi në rrjetet përkatëse

Fillimisht punonjësi mbush fushat e kërkuara me të dhënat përkatëse të certifikatës së pronësisë. Më pas një pjesë e këtyre të dhënave enkriptohen duke përdorur një celës enkriptimi të specifikuar nga punonjësi, përdoruesi apo të vendosur në mënyrë të rastësishme. Më pash, së bashku me të dhënat direkte, pra ato të dhëna që nuk kanë nevojë për enkriptim, krijohet transaksioni. Transaksioni si parametër tjetër merr celësin privat të adresës që do të bëjë lëshimin e certifikatës. Në momentin që certifikata lëshohet për konfirmin, duhet që lëshuesi të presë për faturën e transaksionit. Kjo faturë përfshin të dhëna të nevojshme sic janë adresa e transaksionit, gasi i përdorur, blloku i përfshirë, etj. Ndryshe nga rasti i mësipërm, në fillim bëher lidhja me IPFS, nga ku më pas merren hashet e fotove, dhe këto hashe futen si pjesë e transaksionit.

KAPITULLI 5: ZHVILLIMI

Zhvillimi i një ekosistemi të ngjashëm me atë të lartpërmendur do të ndahet në disa faza, ku në fillim do të krijohen bazat për ndërveprimin me blockchainin, pra do të krijohet kontrata virtuale. Pas kësaj do të krijohet dhe konfigurohen aplikacionet ndërvepruese me kontratën dhe blockchainin. Deri në këtë pikë do të përdorim paisje zhvilluese të ndryshme për të lehtësuar zhvillimin e aplikacionit dhe për të kursyer kohë. Mjedisi i zhvillimit nuk do të përfaqësojë mjedisin përfundimtar. Pasi të jenë kryer testimet e nevojshme, do të krijohen dhe kontejnerizohen nodet e IPFS dhe Blockchain në instanca të ndara nga ofrues të ndryshëm. Në këtë rast për nodet është zgjedhur DigitalOcean, pasi ofron krijim të shpejtë serverash dhe imazhe të ndryshme sistemesh operimi. Për servimin e ndërfaqeve idealisht do të përdornim shërbime si Netlify, CloudFlare Pages ose GitHub pages ose shërbime të tilla të cilat ofrojnë CDN dhe Load Balancing, por në këtë rast do të servirim faqen nga një server i DigitalOcean, ku si CDN do përdorim CloudFlare dhe si WebServer do përdorim NGINX. NGINX do të përdoret dhe si reverse proxy i gateway-t i konfiguruar me SSL për të ridirektuar trafikun nga përdoruesi në blockchain. Kjo sepse API i Geth nuk suporton SSL. Cdo trafik që nuk serviret me SSL duhet të limitohet ndjeshëm. Kjo ose duke lejuar komunikim lokal, ose me një set specifik IP.

5.1. Kontrata Virtuale

5.1.1. Mjedisi i zhvillimit

Për ndërtimin e mjedisit të zhvillimit do të përdorim Ganache-CLI, e cila është pjesë e mjeteve të zhvillimit nga Ethereum. Kjo vjen si paketë në NPM, dhe për ekzekutimin e saj do të përdorim NodeJS. Me një konfigurim të thjeshtë ku përdorim node-http-server për të servirur outputin e ganache-cli.
Komanda që përdorim është “ganache-cli -b 9 -n -u 0 -p 3000”
Këto konfigurime shpjegohen si më poshtë:

-b 9	Block Time - Koha për të përfshirë transaksionet në bllok
-n	Mbyll llogaritë me celës privat
-u 9	Hap llogarinë e parë
-p 3000	Numri i portës së RPC (Remote Procedure Call)

Tabela 5.1 - Lista e opsioneve të ganache-cli

```
Ganache CLI v6.10.1 (ganache-core: 2.11.2)

Available Accounts
==================
(0) 0x17e0372515EA207875B042442eb35B211DF9E9A0 (100 ETH)
(1) 0xc5EfB392d5ca3b018d5F3B366b20e1E14b0aeaD2 (100 ETH)
(2) 0x55acA74414fcA7d5bFA6B6373C096d76cDBd6e69 (100 ETH)
(3) 0x1e782a1a70d6dBa3996f517DA30745839AEe9421 (100 ETH)
(4) 0xB629a9C0e3B653c885573fE776b54aB8323FD784 (100 ETH)
(5) 0x41B1795717E71f3976ae20A681E1989345f77A9b (100 ETH)
(6) 0x5DcE9B187280782cEcf90080efEbdB7Dc3779B97 (100 ETH)
(7) 0x82326575985aBBF31cfA661377f08568E0cF88e7 (100 ETH)
(8) 0x34160468c19dC4990d26C69E0Eb20aab99f857d8 (100 ETH)
(9) 0xb465c5A2A637afC9D64AB489ceCFe39CBf1B4f8D (100 ETH)

Private Keys
==================
(0) 0x37caf2a99b1adc224af59fa849f80f3c8b690344fedd43df7e937ece2e132052
(1) 0x20073b3222afb8afc0eeacc0bcb4c9f03ccbcc4d4579981e97e5950594c083f5
(2) 0x9c6c56b3c3ae56d08a981044bb7e0c2d5d8467971e6fa74a58add63b11b5bfed
(3) 0x7bce0ae748e07b5347c2f329db532231ceb680c0f2c1630d84697abc54612893
(4) 0xa4f7910f6d1e136576a42d074c974db4f479c98f7bf2a2b12a621ed799a9c171
(5) 0xa509c7c96d44218a8598b902a63ba2b75d3cafab6425e7359adf31b7a13b9bb1
(6) 0xcb935e9917a5f52d656d7cd4cbe79e94889a079bfd69137197e4a009d933701d
(7) 0x1c490449f3d10a057942e1b7b794afc27321463c8e7ed41c4920a628c5162353
(8) 0x6c90e792ac2be76720b75fb3c11a50a8e5259ca40250bfb885908e5ed4162d6d
(9) 0x5d25eb1585381f7a8cc7066095a7217b2850d7778afe3d8947172579f75cbe53

HD Wallet
==================
Mnemonic:      reward concert cost second unit crop adult fence possible ride series burden
Base HD Path:  m/44'/60'/0'/0/{account_index}

Gas Price
==================
20000000000

Gas Limit
==================
6721975

Call Gas Limit
==================
9007199254740991

Listening on 127.0.0.1:3000
```

5.1 - Output i ganache-cli

Pasi kryhet ekzekutimi i komandës së mësipërme, shfaqen të dhënat si në figurën 5.1. Na nevojiten kryesisht vetëm llogaria e parë me indeks 0, celësi i saj, dhe duhet treguar kujdes me gas price dhe gas limit, pasi firmosja e transaksionit duhet të jenë në përputhje me këto të dhëna.

Si IDE për zhvillimin e kontratës do të përdoret REMIX IDE, e cila lidhet me mjedisin e testimit nëpërmjet Web3 Provider, ku adresa është në formatin IP:PORT ose DOMAIN:PORT.

5.2 - Fusha për konfigurimin e Web3

Kjo në rastin tonë është si në figurën 5.2

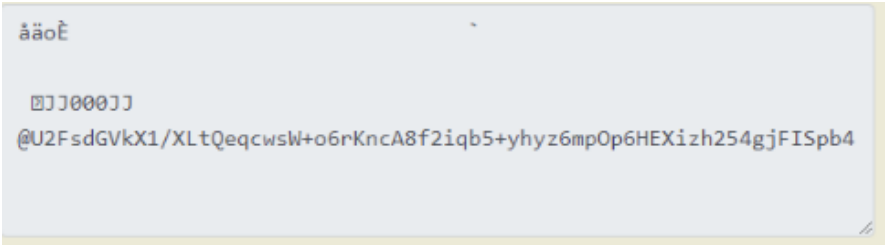
5.1.2. Kontrata Virtuale

Kontrata virtuale do ndahet në dy pjesë. Pjesa e parë do të merret me ruajtjen dhe futjen e certifikatave në sistem, ndërsa pjesa e dytë do të merret me përcaktimin e roleve sipas adresave. Fillimisht të dhënat e certifikatës do të kenë një strukturë të tillë:

- string rawData
- string encData
- uint256 created
- State state
- string photoHash

Ku state është një enumerator që merr vlerat PENDING, APPROVED, DECLINED.Vlera që vendoset në fillim është PENDING. Funksioni tjetër është shtimi i certifikatës

Gjithashtu, duke qënë se blockchaini është komplet transparent, duket të sigurohemi që të dhënat e përdoruesit të mos ekspozohen ndaj publikut, por duke ruajtur integritetitn dhe verifikueshmërinë e tyre. Kjo në vetvete është një sfidë shumë e madhe. E vetmja mënyrë është enkriptimi i të dhënave, mirëpo ky enkripim nuk duhet të kryhet në kontratën virtuale, pasi të dhënat janë ekspozuar në transaksionin fillestar, dhe ndonëse ruhen të enkriptuara në blockchain, në momentin e transferimit ato mbeten publike. Një pohim i tillë është i verifikueshëm lehtë. Si provë lëshojmë një seri të dhënash, si psh ID e një personi “JJ000JJ”. si më poshtë.



5.3 - Ekspozimi i të dhënave

Sic shikohet në figurën 5.3, të dhënat e përdoruesit janë lehtësisht të dallueshme. Mënyra më e mirë do të ishte enkriptimi i të dhënave që para lëshimit, pra që në programin e personit që lëshon certifikata, dhe më pas paisja e pronarit me celësin e enkriptimit. Kjo siguron që pa paisjen e këtij celësi të enkriptimit, marrja e të dhënave do të bëhet e pamundur.

.

```
function addCert(PropertyID[], publicData[], encryptedData[], State[], photoHash[]) {
        if (isIssuer)
                return;
        for (i = 0, i < totalCerts, i++) {
                if(certificateExists)
                        return;
                certificates[PropertyID[i]] = {
                        PropertyID[i],
                        publicData[i],
                        encryptedData[i],
                        State[i],
                        photoHash[i]
                }
                totalCerts+1;
        }
}
```

5.4 - Pseudokod i shtimit të certifikatës

Sic shihet në figurën 5.4 fillimisht kontrollohet nëse adresa ka akses për të kryer veprimin e specifikuar. Pastaj merret vektori i certifikatave të lëshuara dhe u shtohet totalit të certifikatave që janë aktualisht të lëshuara. Pjesa e vektorit është në kuadër të paralelizimit. Më vonë do të vendosim një balancë mes numrit maksimal të certifikatave që mund të lëshohen njëkohësisht. Në rast se gjendet një certifikatë me të njejtën ID atëherë bëhet ndalimi i ekzekutimit.
Pjesa tjetër përfshin funksionet që i takojnë roleve. Rregulli kryesor dhe më i rëndësishëm është kontrolli i lëshuesit të certifikatës. Duhet të sigurohemi që adresa është e autorizuar të lëshojë certifikata. Blockchain funksionon në mënyrë të tillë që cdo adresë e mundshme mund të bëjë transaksione në rrjet si dhe në kontratën virtuale (ose smart contract), dhe për të verifikuar nëse transaksioni i bërë nga një adresë është apo jo autentik, përdorim të ashtëquajturin "private key" apo kodi privat. Ky kod në bashkëpunim me adresën e blockchain na siguron 100% që adresa në fjalë është e autorizuar të kryejë këtë transaksion. Kjo quhet ndryshe si "firmosje transaksioni".
Ngjashëm si certifikata kemi dhe entitetin, me një strukturë si më poshtë:

- address issuerAddress
- string issuerName
- bool issuerState

Në këtë rast kemi një konstruktor, i cili vendos adresen master në lëshimin e kontratës, si dhe përcakton dërguesin e kontratës si root.

```
function addIssuer(issuerAddress, name) {
        if(issuer OR master) {
                updateIssuersList(issuerAddress, name, valid);
                totalIssuers++;
        }
}
```

5.5 - Pseudokod i shtimit të lëshuesit

Funksioni më sipër bën të mundur shtimin e entiteteve dhe lëshuesve të rinj. Mjafton vetëm adresa e lëshuesit, dhe ai bëhet pjesë e adresave që mund të lëshojnë certifikata.

```
function switchIssuer(address) {
        if (isMaster(txSender)) {
                forEach(Issuer)
                        if (issuerFound) {
                                if (issuerValid) {
                                        disableIssuer;
                                } else {
                                        enableIssuer;
                                }
                        }
                }
        }
}
```

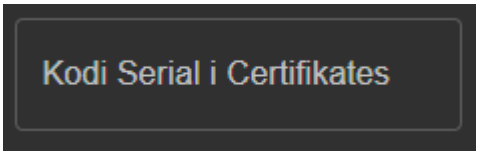
5.6 - Pseudokod i aktivizimit/caktivizimit të lëshuesit

Kjo pjesë bën deaktivizimin e lëshuesit në raste kur nevojitet. I vetmi që mund ta ekzekutojë këtë funksion është entiteti që ka adresën master.

5.2. Ndërfaqet

Për ndërtimin e ndërfaqeve do të përdoret ReactJS.
"React - Një librari JavaScript për ndërtimin e ndërfaqeve" [16]
Për instalimin e ReactJS kërkohet NodeJS dhe NPM. Instalimi varion sipas sistemeve të operimit. Pas kemi instaluar këto programe, instalojmë paketën create-react-app e cila bën të mundur krijimin e një mjedisi reacti, ku përfshihet kompilimi i kodit dhe webserveri i testimit. Me komandën "create-react-app <emri>" bëhet inicializimi i aplikacionit react. Më pas për hapjen e serverit të testimit përdorim "npm run start", i cili hap projektin fillestar të react në adresën "localhost:3000". Ndërtimi i ndërfaqes varion gjerësisht, por për të ruajtur një standard do të përdorim Material-UI, e cila është e thjeshtë në përdorim, dhe krijon UI të përdorshme pa shumë punë.

```
<TextField
        type='text'
        label='Kodi Serial i Certifikates'
        name='propertyID'
        variant='outlined'
        onChange={(e) => setPropertyId(e.target.value)}
/>
```



5.7 - Kodi në react dhe outputi vizual

Kodi në figurën më sipër është një shembull për krijimin e një fushe teksti. Kjo për sa i përket anës vizuale, pasi me këtë mënyrë do të krijohen dhe fushat e tjera, si tek verifikuesi, ashtu edhe tek lëshuesi.

Për sa i përket lidhjes me kontratën virtuale duhet të përdorim librarinë “web3”. Kjo librari bën të mundut lidhjen e ndërfaqes me gatewayn/portën e specifikuar. Pasi kryhet ndërtimi i ndërfaqeve, bëhet kompilimi i tyre me “npm run build”, ku krijohen file specifike në javascript dhe kryhen optimizime nga procesi i kompilimit. Në fund kemi një folder build, të cilin e transferojmë në webserverin tonë. Për momentin në linuxe transferojmë tek vendndodhja “/var/www/html”.

5.2.1. Ndërfaqja e Verifikuesit

Fillimisht specifkojmë disa konfigurime, të cilat janë adresa e kontratës, gateway i blockchainit dhe IPFS. Më pas duhet ABI i kontratës, i cili merret gjatë kompilimit të kontratës në bytecode. Kjo bën të mundur regjistrimin e metodave të kontratës si metoda të javascriptit.

```
function ConnectC(id) {
  // Inicializo Web3
  const web3 = new Web3(new Web3.providers.HttpProvider(network));

  // Vendos Llogarine
  web3.eth.defaultAccount = web3.eth.accounts[0];

  // Set the Contract
  let contract = new web3.eth.Contract(contractAbi, contractAddress);
  console.log(ascii_to_hexa(id));

  const data = contract.methods.certs(ascii_to_hexa(id)).call();
  //console.log(data);
  return data;
}
```

5.8 - Kodi përgjegjës për lidhjen me networkun

Kodi në figurën 5.8 bën konvertimin e ID së pronës në hexadeximal dhe e dërgon në blockchain. Më pas kthehen vlerat e specifikuara nga kontrata. “Certs” kthen të dhënat sipas strukturës së specifikuar në kontratën virtuale. Struktura e të dhënave publike është

- Lloji i truallit
- Qyteti
- Drejtori
- Data e Lëshimit
- Statusi
- Hashi i fotos

```
> ConnectC("HP33443")
<- (5) ['Truall;Tirane;Klajdi H.;1632752066', 'QSxiRHpMqjG
   KK7q3Ev1tJCn9BneQa8hL1jWDgED2D35pQJtqr8Pd'] i
     0: "Truall;Tirane;Klajdi H.;1632752066"
     1: "QSxiRHpMqjG2v563pojA8wmJn6Rin8a6Q67bITgmHSc="
     2: 1632752066
     3: 1
     4: "QmWAxKK7q3Ev1tJCn9BneQa8hL1jWDgED2D35pQJtqr8Pd"
```

5.9 - Të dhënat që merren nga kontrata

Funksioni ConnectC i deklaruar më lart kthen fushat si në figurën më sipër.

Për dekriptimin e të dhënave përdorim AES.
AES është një mënyrë enkriptimi simetrike për enkriptimin e të dhënave sensitive ose konfidenciale [17]. Është përdorur nga qeveria Amerikane për ruajtjen e informacionit të klasifikuar. AES përfshin tre algoritme, AES-128, AES-192 dhe AES-256. Këto numra përfaqësojnë gjatësinë e celësit të enkriptimit (në bits). Zakonisht përdoret gjatësia e celësit 256-bit, pasi dhe sulmet brute-force janë joefektive. Nga ana tjetër 128-bit mund të përdoret për veprime më të shpejta

.

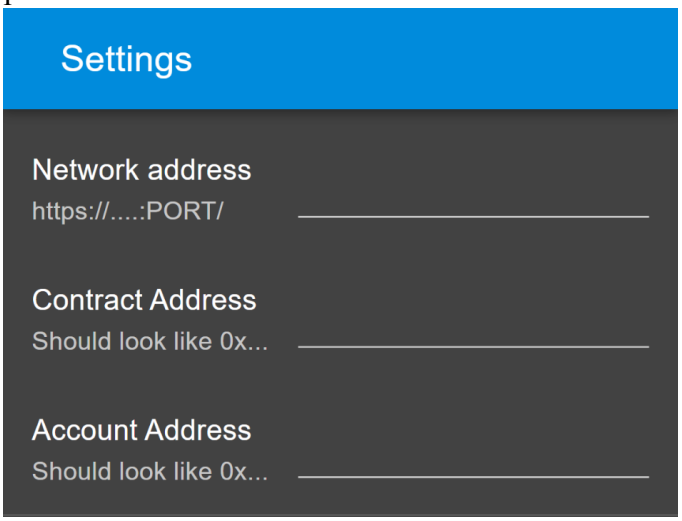
```
AES.decrypt(encData, pass).toString(Utf8)
```

Ky rresht bën dekriptimin e të dhënave pasi specifikohet celësi i dekriptimit. Më pas bëhet thirrja në IPFS, merret imazhi, dekriptohet dhe konvertohet në imazh në format base64. Në fund ky imazh shfaqet në ekranin e përdoruesit, ose përdoret në gjenerimin e një kopje PDF. Njëkohësisht shfaqen dhe të dhënat e dekriptuara, të cilat janë në formatin e mëposhtëm:

- Emrat/Mbiemrat e pronarëve
- ID e pronarëve
- Pjesa e cdo pronari
- Numri i pasurisë
- Zona kadastrale
- Sipërfaqja
- Vlera e pasurisë
- Adresa e plotë e pasurisë

5.2.2. Ndërfaqja e Lëshuesit

Gjatë konfigurimit të kësaj ndërfaqje bëhet vendosja e portës/gatwayst, adresa e lëshuesit që do të përdoret si dhe adresa e kontratës.



5.10 - Shembull i faqes së konfigurimit

Fillimisht një punonjës bën futjen e informacionit të disa certifikatave. Ky informacion ruhet në memorien lokale të browserit. Këtu kemi një limit prej maksimumi 5MB, pasi ky është maksimumi që mund të ruhet në memorien lokale të browserit. Kjo bëhet me anë të metodave të "localStorage" sic janë "getData()" dhe "setData(). Të dhënat ruhen në format JSON.

```
function StoreData(CID, data, pass, photo) {
  let certsBuffer = JSON.parse(localStorage.getItem('certsBuffer')) ? JSON.parse(localStorage.getItem('certsBuffer')) : [];
  certsBuffer.push([CID, data, pass, photo]);
  localStorage.setItem('certsBuffer', JSON.stringify(certsBuffer));
  console.log(JSON.parse(localStorage.getItem('certsBuffer')));
}
```

5.11 - Përbërja e transaksionit para firmosjes

Në këtë fazë imazhi i ngarkuar konvertohet dhe një herë në base64, kjo për të eleminuar karakteret e padëshirueshme sic jane pikëpresjet ose slashet. Pasi përfundohet me plotësimin e certifikatave vijohet me eksportimin e tyre në një dokument JSON. Kjo bëhet për të mundësuar transferimin dhe firmosjen e të dhënave në një kompjuter tjetër. Për importimin e dokumentit krijohet një vendndodhje tjetër në po të njejtën ndërfaqe. Në këtë fazë bëhet importimi i të dhënave nga dokumenti JSON, ndarja e tyre në publike dhe private. Veprimi i parë që bëhet pas dërgimit të kontratave në rrjet është enkriptimi i fotove dhe dërgimi i tyre në IPFS. Pritet që dërgimi të përfundojë dhe të kthehen hashet për cdo foto. Pasi kjo përfundon vijohet me enkriptimin e të dhënave të tjera. Përdoruesit i kërkohet të vendosë celësin privat të adresës me të cilën do firmosen transaksionet. Në këtë moment krijohet transaksioni.

```
let tx = {
   from: sendAddress,
   to: contractAddress,
   gasPrice: 1000000000,
   gas: 15000000,
   data: addCertABI
};
```

5.12 - Përbërja e transaksionit para firmosjes

Transaksioni i ngjan kodit të mësipërm, dhe është ky set të dhënash që firmoset nga celësi privat i adresës së lëshuesit.

```
▼Object
   blockHash: "0xcfc13e5e576aea5f87abb98374697840933abda6baced0fa1bb67d1821c59482"
   blockNumber: 369212
   contractAddress: null
   cumulativeGasUsed: 344048
   from: "0x469a27b7315e16b16800e716359a647cb2f91904"
   gasUsed: 344048
  ▶logs: []
   logsBloom: "0x00000000000000000000000000000000000000000000000000000000000000000…
   status: true
   to: "0x008421d0e4bfbf2aa49e5d3cc3924a7ee2b250e4"
   transactionHash: "0x35bb41aeb219a224d3fe824bcbd99be2d27e4c8354ecf6b054162f914d1b…
   transactionIndex: 0
   type: "0x0"
```

5.13 - Receta e transaksionit

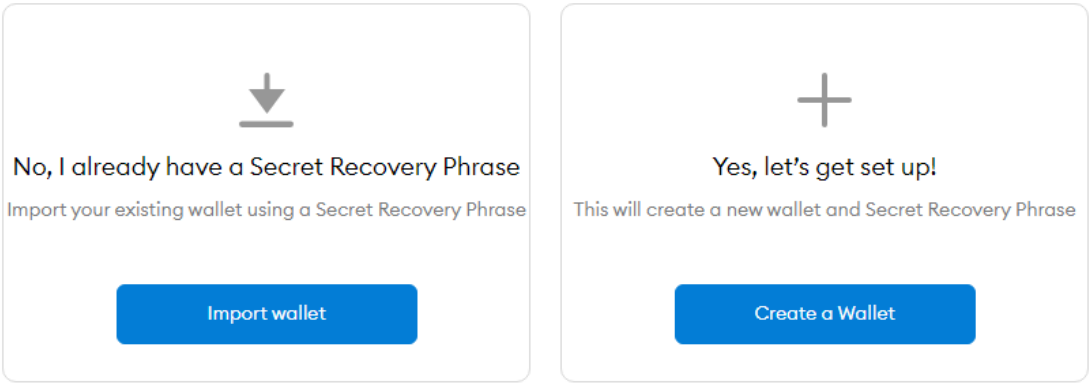
Në fund të këtij veprimi përdoruesit i shfaqet një recetë transaksioni.Kjo recetë është e ngjashme me foton [FOTO] dhe përmban të dhëna si:

- Hashi i bllokut
- Numri i bllokut
- Gasi i përdorur
- Adresa e dërguar
- Hashi transaksionit

Ky proces tregon mbarimin e procesit të lëshimit të certifikatës.

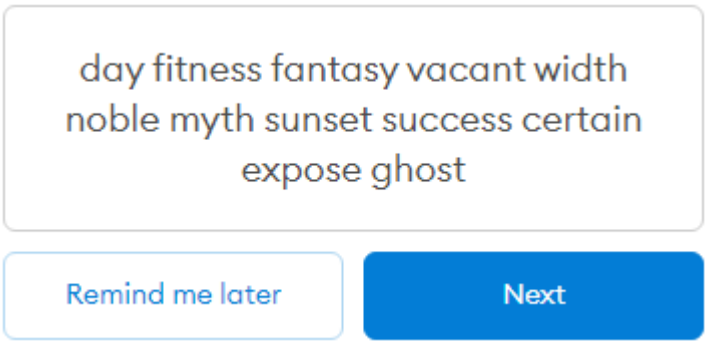
5.3. Krijimi i Adresave

Për krijimin e adresave mund të përdoren disa mënyra, një nga ato është nëpërmjet ndërfaqes së MetaMask. MetaMask është një kripto-portofol dhe portë për aplikacionet në blockchain. Ndonëse mund të përdoret për verifikimin e certifikatave, kryesisht do ta përdorim vetëm për lëshimin e kontratës virtuale dhe krijimin e adresave të nevojshme. Kjo për të vetmen arsye se synimi jonë është krijimi i një aplikacioni sa më të lehtë për tu përdorur. MetaMask vjen si shtesë e Chrome ose Edge, dhe pas instalimit kemi ndërfaqen e tillë:



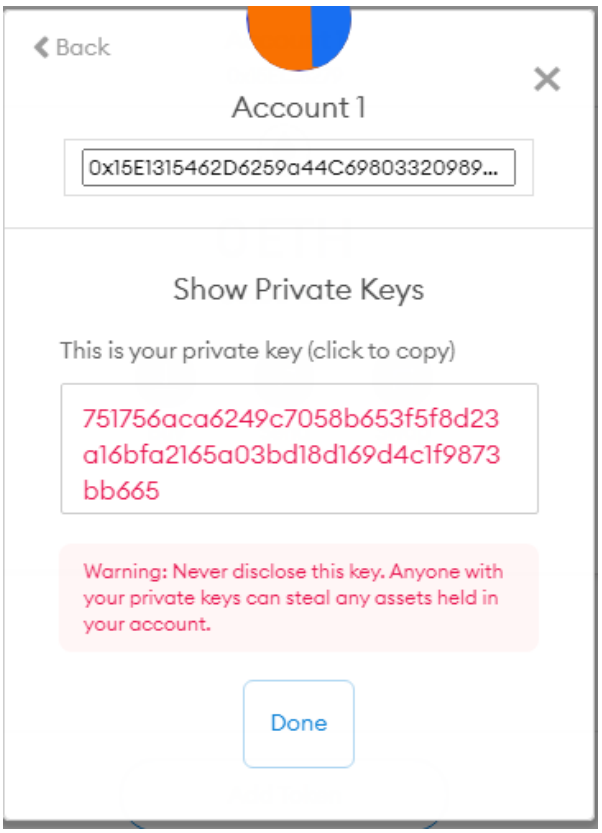
5.14 - Fillim i krijimit të adresës

Në rastin tonë krijojmë një portofol, ku pas klikimit të butonit pranojmë kushtet dhe termat dhe më pas plotësojmë passëordin. Proces i rëndësishëm është ruajtja e fjalëve të mëposhtme të cilat do të përdoren për të rimarrë akses në adresë në rast se humbet celësi privat.



5.15 - Frazat mnemonike të llogarisë

Pasi kemi ruajtur këtë frazë (ky është opsional, pasi problemet që zgjidh ky hap merren parasysh në zgjidhjet e mësipërme) shohim ndërfaqen kryesore. Për të marrë celësin privat shkojmë tek opsionet > “Account details” > “Export Private Key”.



5.16 - Celësi publik dhe celësi privat

Sic shohim në foton 5.15, ky është celësi privat, i cili në kombinim me adresën do të përdoren për firmosjen dhe lëshimin e transaksioneve në rrjet.

5.4. Ndërtimi i rrjetit Blockchain

Për krijimin e rrjetit blockchain do të përdorim "geth" në një server Ubuntu 18.04.
Fillimisht ekzekutohen komandat e mëposhtme:

- add-apt-repository -y ppa:ethereum/ethereum
- apt install ethereum
- adduser node1

Fillimisht shtohet repozitori i ethereum, bëhet përditësimi i paketave dhe instalimi i ethereum. Me instalimin e ethereum na vijnë dy komanda të nevojshme të cilat janë "puppeth" dhe "geth". E para përdoret për gjenerimin e bllokut gjenezë, dhe e dyta për inicializimin e anëtarit të blockchain.
Për krijimin fillestar të networkut kërkohet të krijohet blloku gjenezë, i cili krijohet me komandën puppeth.

```
root@Node1:~# puppeth
+-----------------------------------------------------------+
| Welcome to puppeth, your Ethereum private network manager |
|                                                           |
| This tool lets you create a new Ethereum network down to  |
| the genesis block, bootnodes, miners and ethstats servers |
| without the hassle that it would normally entail.         |
|                                                           |
| Puppeth uses SSH to dial in to remote servers, and builds |
| its network components out of Docker containers using the |
| docker-compose toolset.                                   |
+-----------------------------------------------------------+

Please specify a network name to administer (no spaces, hyphens or capital letters please)
> 
```

5.17 - Output i komandës puppeth

Fotoja më sipër tregon outputin e komandës që sapo ekzekutuam.

```
INFO [09-27|15:40:06.812] Administering Ethereum network           name=rems
WARN [09-27|15:40:06.812] No previous configurations found         path=/root/.puppeth/rems

What would you like to do? (default = stats)
 1. Show network stats
 2. Configure new genesis
 3. Track new remote server
 4. Deploy network components
>
```

5.18 - Opsionet fillestare të puppeth

Më pas kemi selektimin e veprimit që duam të bëjmë. Në rastin tonë është konfigurimi i bllokut gjenezë. Pas kësaj zgjedhim konsensusin “Clique”, dhe për kohën e bllokut specifikojmë 15 sekonda. Pas kësaj është e detyrueshme të specifikojmë adresën e validuesit të parë, kjo bëhet dy herë, një herë për specifikimin e validuesit dhe herën e dytë për ti dhënë gas asaj adrese. Vijojmë me specifikimin e një numri random, i cili do të përfaqsojë ID e networkut.

```
Which consensus engine to use? (default = clique)
 1. Ethash - proof-of-work
 2. Clique - proof-of-authority
> 2

How many seconds should blocks take? (default = 15)
> 15

Which accounts are allowed to seal? (mandatory at least one)
> 0x193D9169748Bf8C4D1A1B605CB4cc0e33d5b34fc
> 0x

Which accounts should be pre-funded? (advisable at least one)
> 0x193D9169748Bf8C4D1A1B605CB4cc0e33d5b34fc
> 0x

Should the precompile-addresses (0x1 .. 0xff) be pre-funded with 1 wei? (advisable yes)
> yes

Specify your chain/network ID if you want an explicit one (default = random)
> 0369
INFO [09-27|15:46:46.080] Configured new genesis block

What would you like to do? (default = stats)
 1. Show network stats
 2. Manage existing genesis
 3. Track new remote server
 4. Deploy network components
>
```

5.19 - Përfundimet e krijimit të gjenezës

Në fund të procedurës kemi ekranin si më sipër. E heqim ekranin dhe shohim që në vendndodhjen aktuale është gjeneruar një dokument. Ky dokument përmban të dhënat e bllokut gjenezë. Disa fusha në këtë dokument janë:

chainId	ID e rrjetit blockchain
period	Perioda e gjenerimit të blloqeve
epoch	Numri i blloqeve më të fundit që validohen
timestamp	Koha e krijimit
gasLimit	Limiti i gasit

Tabela 5.2 - Lista e konfigurimeve të bllokut gjenezë

Për arsye se geth nuk jep mundësi specifikimi të gasLimit, e ndryshojmë atë direkt në dokument dhe e vendosim në përputhje me konfigurimet e mësipërme të ndërfaqeve. GasLimit e vendosëm 15,000,000, dhe në dokument e specifikojmë si 0xe4e1c0. Kjo pasi blloku gjenezë pranon vetëm vlera në heksadecimal.

Pasi kemi konfiguruar bllokun gjenezë, vazhdojmë me inicializimin e networkut. Për këtë përdorim komandën e mëposhtme:
geth --datadir node/ init rems.json

```
node1@Node1:~$ geth --datadir node/ init rems.json
INFO [09-27|16:15:17.278] Maximum peer count                       ETH=50 LES=0 total=50
INFO [09-27|16:15:17.278] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such
ctory"
WARN [09-27|16:15:17.279] Sanitizing cache to Go's GC limits       provided=1024 updated=328
INFO [09-27|16:15:17.279] Set global gas cap                       cap=50,000,000
INFO [09-27|16:15:17.280] Allocated cache and file handles         database=/home/node1/node/geth/chaindata
iB handles=16
INFO [09-27|16:15:17.288] Writing custom genesis block
INFO [09-27|16:15:17.317] Persisted trie from memory database      nodes=356 size=50.51KiB time=2.056986ms
ize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [09-27|16:15:17.318] Successfully wrote genesis state         database=chaindata
.54ef9a
INFO [09-27|16:15:17.318] Allocated cache and file handles         database=/home/node1/node/geth/lightchai
6.00MiB handles=16
INFO [09-27|16:15:17.323] Writing custom genesis block
INFO [09-27|16:15:17.337] Persisted trie from memory database      nodes=356 size=50.51KiB time=1.866878ms
ize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [09-27|16:15:17.338] Successfully wrote genesis state         database=lightchaindata
bd69..54ef9a
node1@Node1:~$
```

5.20 - Inicializim i blockchainit

Pas inicializimit vijojmë hapjen e llogarisë me të cilën do të kryejmë validimin, kjo bëhet duke krijuar një dokument në të njejtin vendndodhje ku jemi, dhe si përmbajtje e këtij dokumenti vendosim celësin privat të adresës që specifikuam si validues.
Më pas përdorim komandën
geth account import --datadir node/ priv.key
Ku priv.key është emri i dokumentit që krijuam, dhe node është folderi ku do të ruhen të dhënat dhe blloqet e blockchain. Pasi ekzekutohet komanda duhet të specifikojmë një fjalëkalim për të siguruar adresën.

```
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:
```

5.21 - Kërkesa për mbrojtje adrese

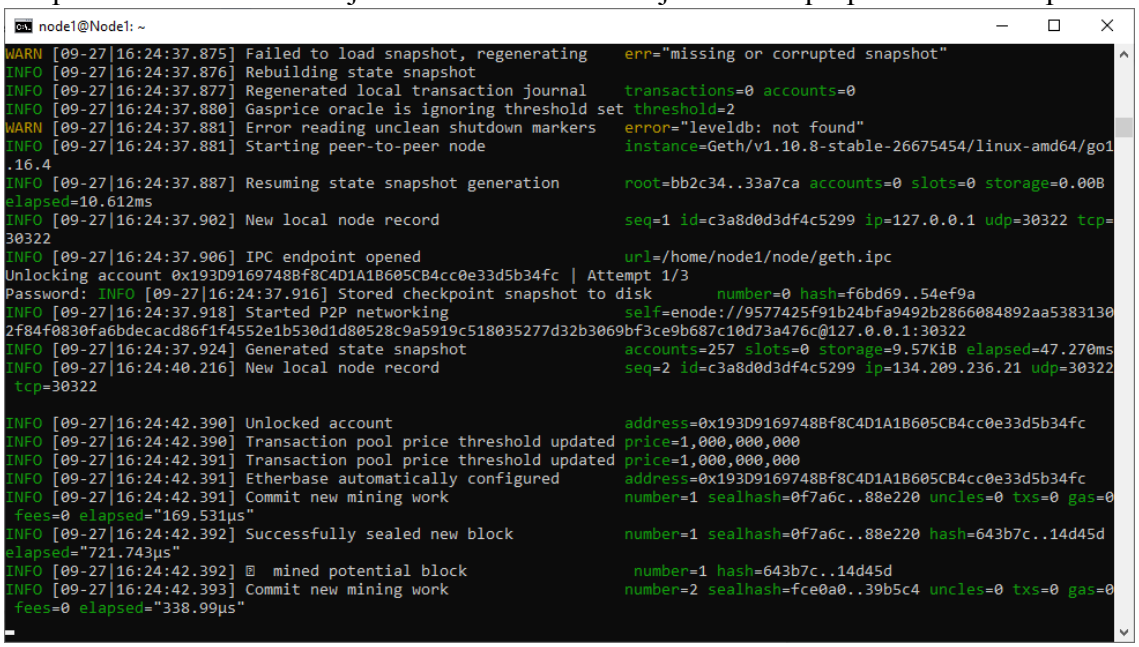
Për startimin e procesit të validimit përdorim komandën e mëposhtme:

“*geth --nousb --datadir=node/ --syncmode 'full' --port 30322 --miner.gasprice 1000000000 --unlock 0x193D9169748Bf8C4D1A1B605CB4cc0e33d5b34fc --mine*”

--nousb	Bllokon komunikimin me USB
--datadir <folder>	Folderi ku do të ruhen të dhënat e blockchainit
--syncmode <mode>	Mënyra e sinkronizimit (full, fast, light).
--port <port>	Porta ku do të aksesohet ky anëtar
--miner.gasprice <wei>	Cmimi për gas që kërkohet për përfshirje transaksioni
--unlock <addr>	Adresa që do të konfirmojë transaksione
--mine	Fillo procesin e konfirmimit të transaksioneve
--bootnodes	Adresa Enode për lidhje me anëtar të tjerë

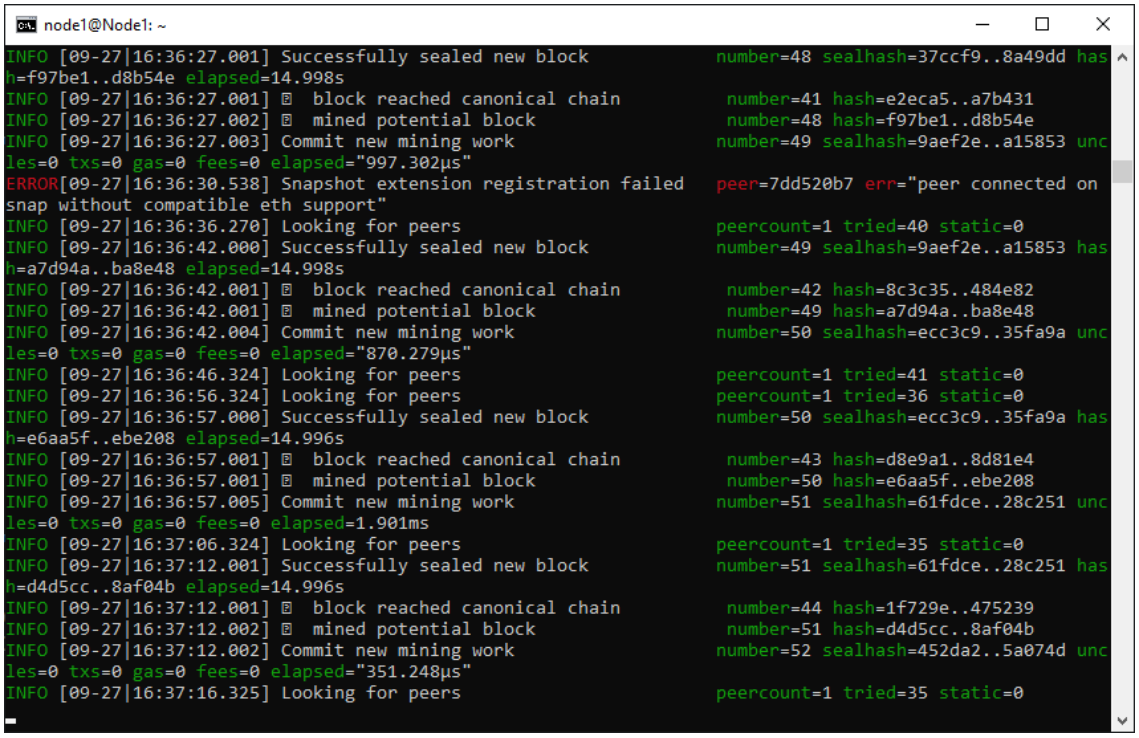
Tabela 5.3 - Lista e opsioneve të geth

Fill pas ekzekutimit të kësaj komande na kërkohet fjalëkalimi që specifikuam më parë.



5.22 - Ekrani fill pas startimit të anëtarit fillestar

Pastaj kemi ekranin si më sipër. Këtu duhet të vëmë re adresën enode.
“enode://9577425f91b24bfa9492b2866084892aa53831302f84f0830fa6bdecacd86f1f4552e1b530d1d80528c9a5919c518035277d32b3069bf3ce9b687c10d73a476c@127.0.0.1:30322”
Kjo adresë bën të mundur lidhjen e nodeve me njëri tjetrin. Vijojmë me të njejtën procedurë në serverin e rradhës, i cili do jetë thjesht anëtar. Komanda për anëtarin është si më poshtë:
geth --nousb --datadir=node/ --syncmode 'full' --port 30321 --miner.gasprice 1000000000 --miner.gastarget 470000000000 --bootnodes “enode://9577425f91b24bfa9492b2866084892aa53831302f84f0830fa6bdecacd86f1f4552e1b530d1d80528c9a5919c518035277d32b3069bf3ce9b687c10d73a476c@127.0.0.1:30322”

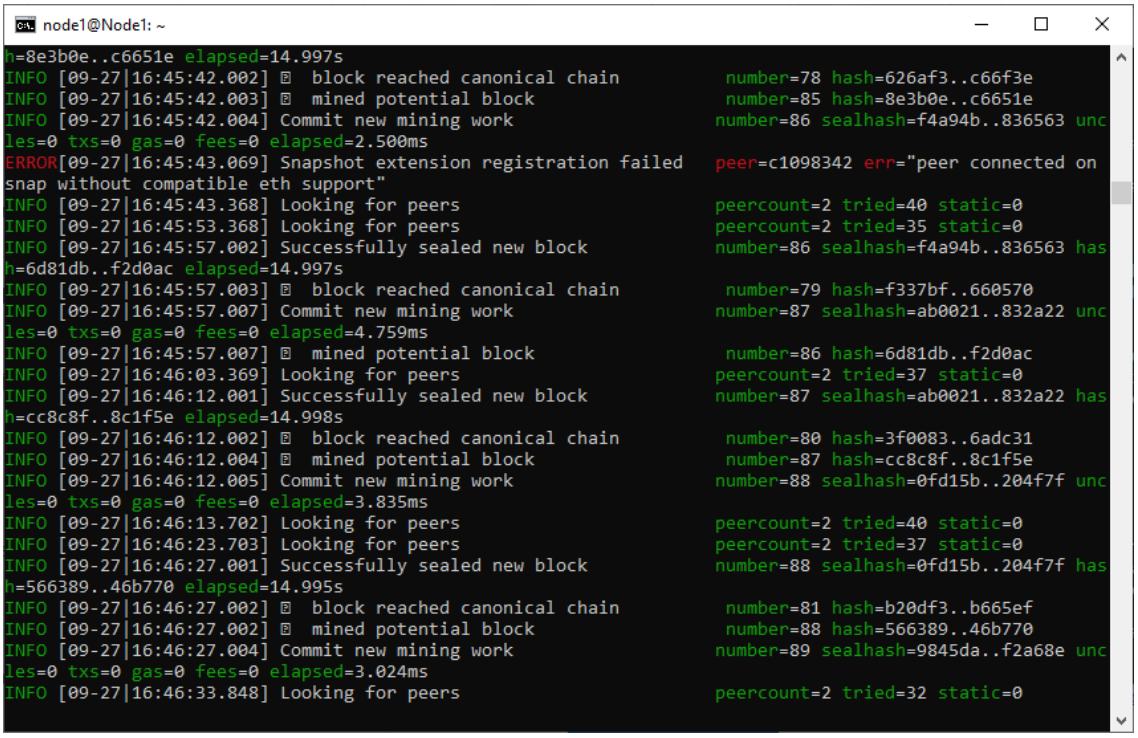


5.23 - Ekrani fill pas startimit të anëtarit të dytë

Sic shihet nga figura, peercount u rrit me 1.
E njejta procedurë bëhet për krijimin e të gjithë anëtarëve të tjerë. Pjesa e mbetur konsiston në krijimin e një gateway. Komanda është si në vijim:
geth --nousb --datadir=node/ --syncmode 'full' --port 30323 --miner.gasprice 1000000000 --http --http.addr '0.0.0.0' --http.port 8503 --http.corsdomain "" --http.vhosts "*" --http.api admin,eth,miner,net,txpool,personal,web3 --bootnodes 'enode://9577425f91b24bfa9492b2866084892aa53831302f84f0830fa6bdecacd86f1f4552e1b530d1d80528c9a5919c518035277d32b3069bf3ce9b687c10d73a476c@127.0.0.1:30322'*

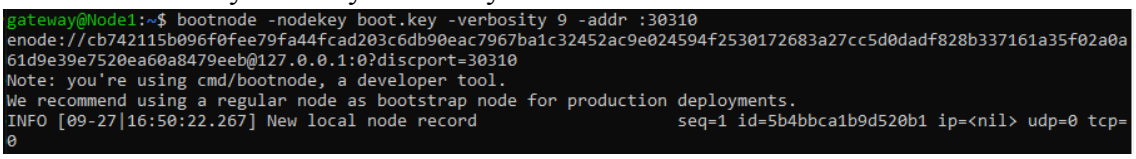
--http	Hap serverin HTTP
--http.addr <IP>	Ndërfaqja ku kalon trafiku, 0.0.0.0 pa limit IP networku
--http.port <port>	Porta ku kalon trafiku
--http.corsdomain <FQDN>	Domainet që lejohen ta aksesojnë
--http.api <api list>	Set i veprimeve që lejohet në HTTP

Tabela 5.4 - Lista e opsioneve të geth (në rastin e gateway)



5.24 - Ekrani fill pas startimit të anëtarit të tretë (portës)

Pasi inicializohet gateway, shohim që numri i peers ka shkuar në 2. Në këtë moment kemi rrjetin e plotë të blockchainit. Në rast se nevojitet mund të krijohet dhe një bootnode, e cila bën lidhjen e nodeve me njëra tjetrën më lehtë me komandat e mëposhtme:
bootnode -genkey boot.key
bootnode -nodekey boot.key -verbosity 9 -addr :30310



5.25 - Krijimi i një anëtari lidhës

Për arsye sigurie, --http.addr do vendoset në rrjet lokal (127.0.0.1) dhe për komunikimin me gatewayn do përdorim reverse proxy.

5.5. Ndërtimi i rrjetit IPFS

Sikurse dhe rrjeti i blockchain, rrjeti i IPFS kërkon disa programe të parainstaluara që të funksionojë. Kërkohet që të kemi instaluar GO Lang. Për instalimin e IPFS shkarkojmë dosjet binare nga faqja zyrtare e IPFS. Versioni aktual është v0.9.1. Pasi shkarkojmë dokumentin e kompresuar, ekstraktojmë dosjet e tjera dhe për linux zhvendosim skedarët aty ku gjenden dhe lista e aplikacioneve të tjera të ekzekutueshme. Më pas kami një strukturë të tillë "/usr/local/bin/ipfs" dhe verifikojmë që ipfs është instaluar me komandën "ipfs version", ku si output marrim versionin e IPFS.

```
root@Node1:~# ipfs init
generating ED25519 keypair...done
peer identity: 12D3KooWAEVUYynwCfMkxXxRCi8G74th3nKH3YLyYVCmXXfrCDwC
initializing IPFS node at /root/.ipfs
to get started, enter:

        ipfs cat /ipfs/QmQPeNsJPyVWPFDVHb77w8G42Fvo15z4bG2X8D2GhfbSXc/readme
```

5.26 - Inicializimi i anëtarit të parë IPFS

Me "ipfs init" inicializohet rrjeti i IPFS dhe kthehet si vlerë identiteti i anëtarit, por momentalisht këtë rrjet duhet ta bëjmë privat. Për këtë përdorim një dokument i cili quhet "swarm.key", ky shërben si identifikues mes anëtarëve të ipfs, dhe është një numër hekzadecimal 64 bajt. Në linux komanda e mëposhtme bën krijimin e këtij dokument sipas kërkesave:

echo -e "/key/swarm/psk/1.0.0/\n/base16/\n`tr -dc 'a-f0-9' < /dev/urandom | head -c64`" > ~/.ipfs/swarm.key

```
root@Node1:~# cat .ipfs/swarm.key
/key/swarm/psk/1.0.0/
/base16/
be0132670ad8a8e9a9de5b5cb5a77b537cfe145b34a37a80f0bd7062077f7210
```

5.27 - Përmbajtja e swarm.key

Përmbajtja e dokumentit të krijuar është si më lart. Ky dokument shpërndahet në cdo anëtar që do të futet në rrjet. Pastaj heqim lidhjet aktuale me networket e tjera me komandën "ipfs bootstrap rm --all". Node fillestar do të shërbejë si node lidhës (bootstrap) për të lidhur nodet private me njëra tjetrën. Në cdo anëtar tjetër ekzekutohet komanda për shtimin e anëtarit lidhës:

ipfs bootstrap add /ip4/<IP>/tcp/4001/ipfs/<ID Anëtarit>

Për të hapur aksesin nga të gjithë ndryshojmë konfigurimin e IPFS:

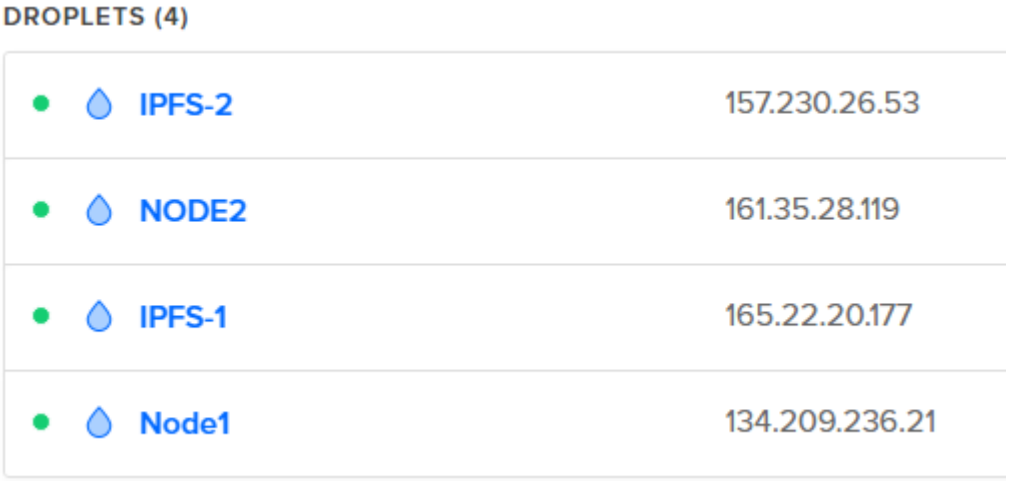
ipfs config Addresses.Gateway /ip4/0.0.0.0/tcp/8080

Kjo nuk duhet ekzekutuar, pasi në rast se nevojitet do të ekspozohet në reverse proxy. Për momentin po përdoret për arsye demonstrimi.

```
root@ubuntu-s-1vcpu-1gb-fra1-01: ~
(Hit ctrl-c again to force-shutdown the daemon.)
root@ubuntu-s-1vcpu-1gb-fra1-01:~# ipfs config Addresses.Gateway /ip4/0.0.0.0/tcp/8080
root@ubuntu-s-1vcpu-1gb-fra1-01:~# ipfs daemon
Initializing daemon...
go-ipfs version: 0.9.1
Repo version: 11
System version: amd64/linux
Golang version: go1.16.6
Swarm is limited to private network of peers with the swarm key
Swarm key fingerprint: 8e2497a8985b06365c4707a63f6c6d76
Swarm listening on /ip4/10.114.0.5/tcp/4001
Swarm listening on /ip4/10.19.0.8/tcp/4001
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/165.22.20.177/tcp/4001
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /p2p-circuit
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/165.22.20.177/tcp/4001
Swarm announcing /ip6/::1/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
WebUI: http://127.0.0.1:5001/webui
Gateway (readonly) server listening on /ip4/0.0.0.0/tcp/8080
Daemon is ready
```

5.28 - Startimi i IPFS

Startojmë IPFS me “ipfs daemon”, dhe kur vizitohet URL-ja me formatin e “http://<IP>:8080/ipfs/<HASH>” shfaqen të dhënat e dokumentit.
Pasi shtojmë dhe anëtarët e tjerë shohim që dokumenti është i aksesueshëm nga cdo anëtar.

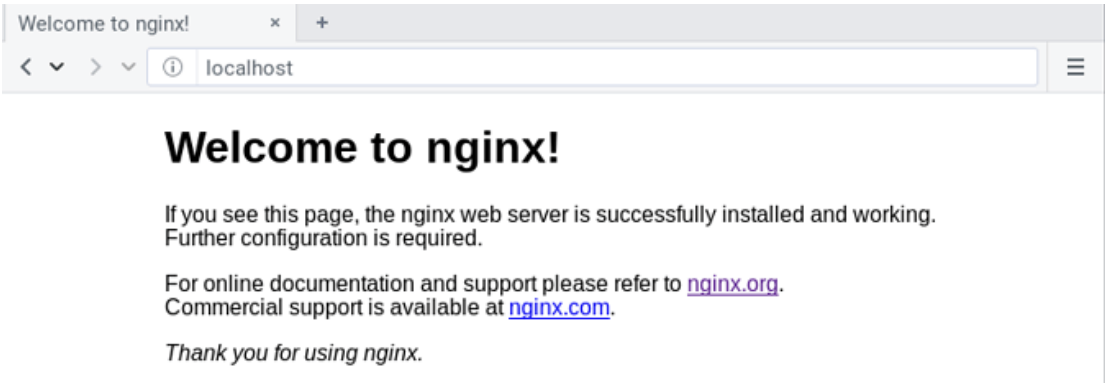


5.29 - Lista e serverave

Fotoja 5.28 tregon listën e serverave të krijuar si pasojë e ngritjes së rrjeteve Blockchain dhe IPFS.

5.6. Sigurimi i portave me Reverse Proxy

Për krijimin e një reverse proxy fillimisht nevojitet një webserver. Në rastin tonë do të përdorim NGINX. Fillimisht instalojmë NGINX me komandën “apt install nginx”. Sapo instalohet nginx, kur vizitojmë IP e serverit shfaqet faqja fillestare e webserverit si në figurën 5.30. Në rast se kemi zhvendosur ndërfaqen tek vendndodhja “/var/www/html” atëherë do të shfaqet ndërfaqja e përdoruesit.



5.30 - Faqja fillestare e nginx

Për konfigurimin e reverse proxy krijojmë një dokument të ri tek /etc/nginx/sites-available/<Emri Portës>

```
server {
        listen 80;
        listen [::]:80;
        listen 443 ssl;
        listen [::]:443 ssl;

    ssl_certificate /etc/letsencrypt/live/<domain>/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/<domain>/privkey.pem;

        server_name <domain> www.<domain>;

        location / {
                proxy_pass http://localhost:8503;
                proxy_set_header Host $host;
                proxy_set_header X-Forwarded-Proto https;
        }
}
```

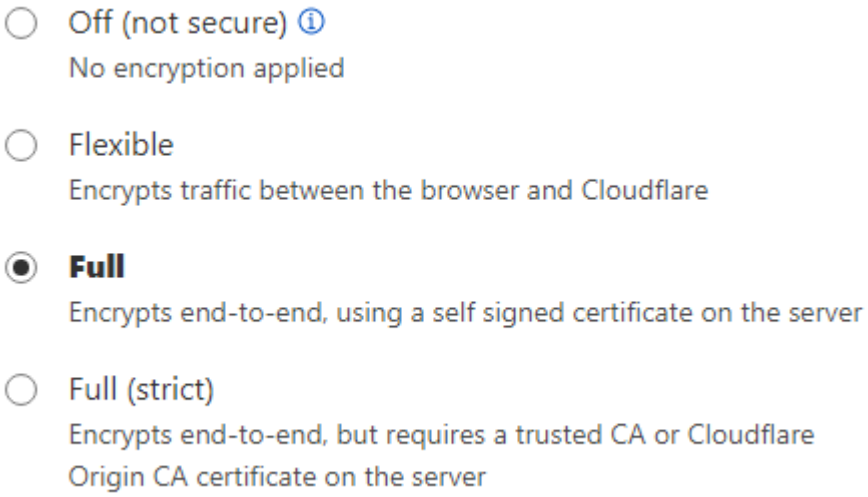
5.31 - Konfigurimi i reverse proxy

Konfigurimi i reverse proxy bëhet si më sipër, ku fillimisht specifikohen portat e jashtme nga ku do të kalojë trafiku (80 dhe 443), pastaj certfikata SSL, domaini nga do të vijë trafiku dhe më pas specifikojmë ku do ta dërgojmë këtë trafik. Në rastin e blockchain do ta dërgojmë në portën lokale 8503. Në disa raste specifike rekomandohet të vendosen header të tjerë sic është "Host" dhe "X-Forwarded-Proto". Për certifikatën SSL mjaftohemi me një "Self-Signed", pasi trafiku do vijë nëpërmjet CloudFlare. Komanda më poshtë bën gjenerimin e dokumenteve të nevojshme
openssl req -x509 -newkey rsa:4096 -nodes -keyout privkey.pem -out fullchain.pem -days 3650 -subj '/CN=<domain>
mv privkey.pem /etc/letsencrypt/live/<domain>
mv fullchain.pem /etc/letsencrypt/live/<domain>

Komanda e parë bën gjenerimin e SSL sipas standardit RSA-4096. Ky një standard kriptografik gjerësisht i përdorshëm në ditët e sotme. Dy komandat e tjera bëjnë zhvendosjen e celësit dhe certifikatës SSL në skedarët përkatës.
Më pas kalojmë në konfigurimin e CloudFlare, ku supozohet se kemi blerë dhe shtuar domainin.

5.32 - Konfigurimi i subdomain në CloudFlare

Kryejmë konfigurimin si më sipër, ku shtojmë një rekord të tipit A me emrin e portës dhe si vlerë japim adresën e portës.



5.33 - Opsionet SSL/TLS në CloudFlare

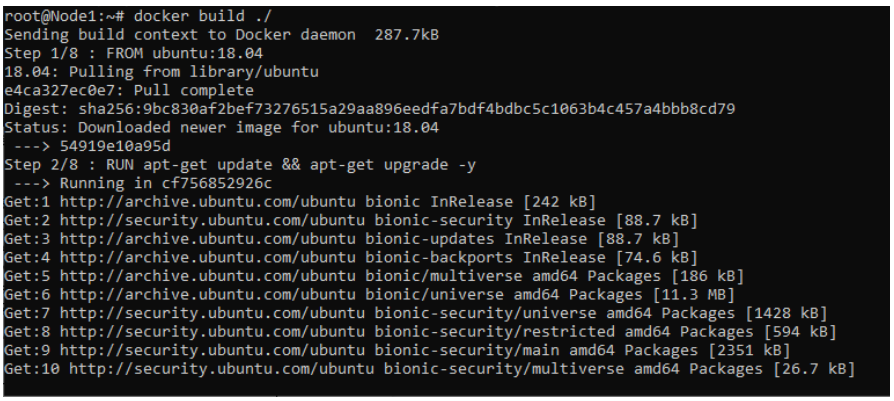
Tek paneli i SSL/TLS zgjedhim Full. Rasti Full (Strict) do të jepte problem pasi certifikata kërkohet të jetë lëshuar nga një autoritet i besueshëm, dhe opsionet e tjera nuk janë të sigurta mjaftueshëm. Në këtë pikë mund të përdorim domainin e konfiguruar për adresën e blockchainit dhe IPFS.

5.7. Dockerizimi / Kontejnerizimi

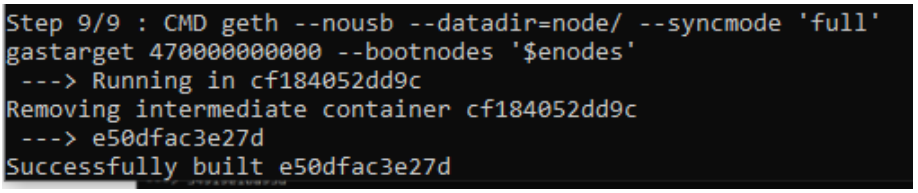
```
1  FROM ubuntu:18.04
2  RUN apt-get update && apt-get upgrade -y
3  RUN apt-get -y install software-properties-common
4  RUN add-apt-repository -y ppa:ethereum/ethereum
5  RUN apt-get -y install ethereum
6  COPY . .
7  ARG enodes
8  EXPOSE 30321
9  RUN geth --datadir node/ init rems.json
10 CMD geth --nousb --datadir=node/ --syncmode 'full' \
11         --port 30321 --miner.gasprice 1000000000 \
12         --miner.gastarget 470000000000 --bootnodes '$enodes'
```

5.34 - Përmbajtja e Dockerfile

Docker fillimisht instalohet me komandën “apt install docker.io”. Për krijimin e docker kërkohet një dokument “Dockerfile”, ku si përmbajtje janë instruktionet që do të ekzekutohen në imazh. Me komandën “FROM” specifikojmë imazhin bazë nga ku do të marrim pjesën mbi të cilën do ndërtojmë rrjetin. Në rastin tonë kemi specifikuar Ubuntu 18.04. Me komandat “RUN” nga rreshti 2 deri tek 5 bëhet instalimi i mjeteve të ethereum. Me “COPY” kopjojmë bllokun gjenezë nga kompjuteri jonë tek imazhi i docker (dockerfile duhet të ekzekutohet në të njejtin vend me bllokun gjenezë). Marrim argument enodes, hapim portën 30321 me “EXPOSE”, inicializojmë anëtarin me rreshtin 9 dhe bëjmë startimin e tij me komandën CMD.
Për të ekzekutuar instruksionet e mësipërme japim komandën “docker build ./” ku i themi dockerit të ndërtojë imazhin në bazë të Dockerfile në vendndodhjen ku jemi.



5.35 - Rezultatet e komandës “docker run”



5.36 - Përfundim i komandës “docker.run”

Në mbarimin e procesit, kemi id e imazhit. Këtë id e përdorim në komandën e startimit të dockerit “docker run -p 30321:30321 e50dfac3e27d” ku bëjmë lidhjen e portës së jashtme me atë të docker (port mapping).
Pa pasur nevojë për konfigurime të tepërta, ky Dockerfile mund tu jepet të tjerëve dhe me 2 komandat “docker build” dhe “docker run” kemi një anëtar të ri në rrjet.

KAPITULLI 6: MATJE DHE REZULTATE

Një tjetër fakt i rëndësishëm është se cdo transaksion kushton dhe merr taksa (ndryshe quhen blockchain fees), të cilat përdoren për llogaritje kompjutacionale nga EVM dhe nga rrjeti i blockchain. Këto taksa nuk janë fikse në blockchain publik, dhe variojnë në bazë të shumë faktorëve globalë. Së fundmi mund të përmendim infulencimin e Elon Musk, presidentit amerikan Joe Biden, rritjen e kërkesës për përdormin e blockchain, vështirësia e minimit, ngarkesa e rrjetit, dhe shumë faktorë të tjerë që janë të pamundur për tu analizuar. E njejta gjë vlen edhe për kohën e konfirmimit. Taksa është proporcionale me madhësinë e të dhënave që shtohen në rrjet, matet me GAS dhe duke marrë parasysh një transaksion prej 200,000 gas në rrjetin Ethereum, kryejmë llogaritjet si më poshtë:

1 GAS = 23.1 GWei
Ku në GWei është cmimi i një gasi, dhe është dinamik, në varësi të faktorëve globalë.

23.1 GWei * 200,000 GAS = 4,620,000 GWei

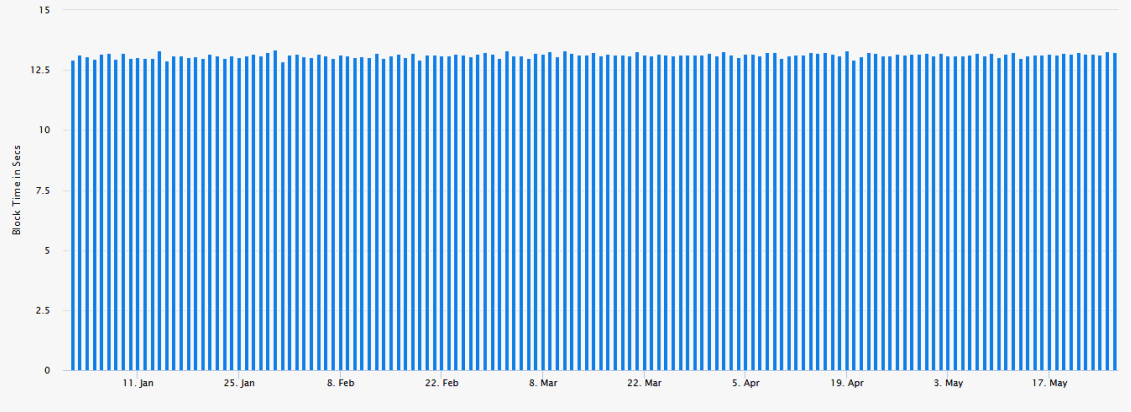
1 GWei = 1 * 10^{-9} ETH
4,620,000 = 0.00462 ETH

0.00462 ETH = 12.9 USD

Ku në momentin e llogaritjeve 1 ETH = 2782.8 USD

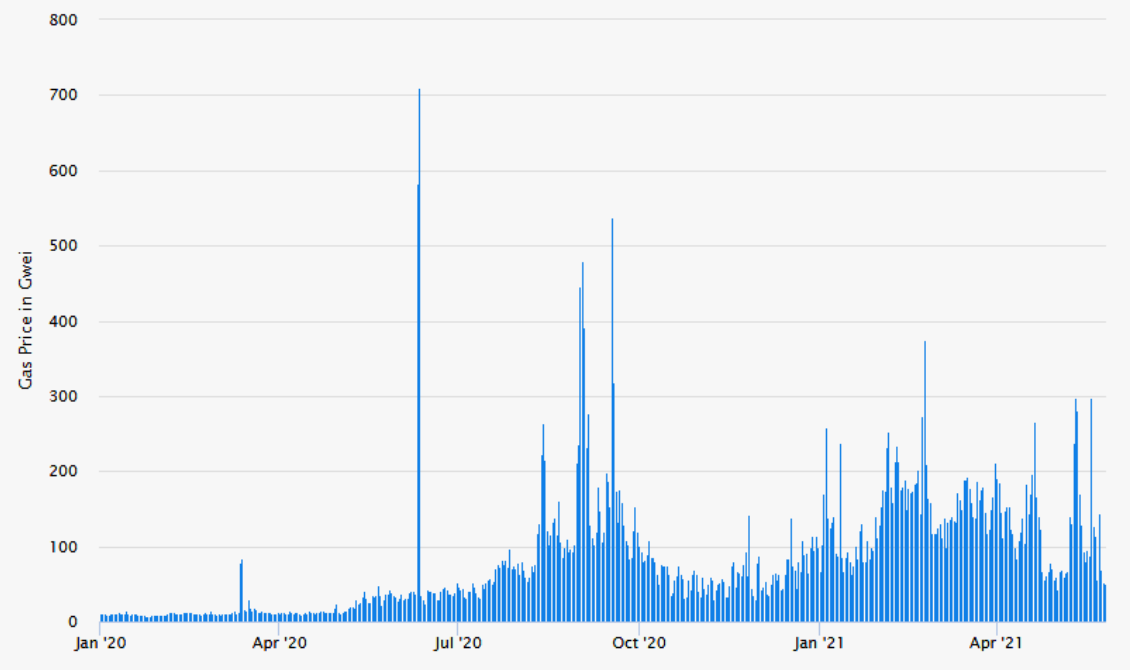
Outcome	
% e 200 blloqeve të fundit që pranojnë këtë cmim gasi	86
Mesatarja e Konfirmimit (Blloqe)	2.1
Mesatarje e konfirmimit (Sekonda)	32
Taksa e rrjetit (ETH)	0.00462
Taksa e rrjetit (USD)	$12.9

Tabela 6.1 - Përmbledhje e Transaksionit

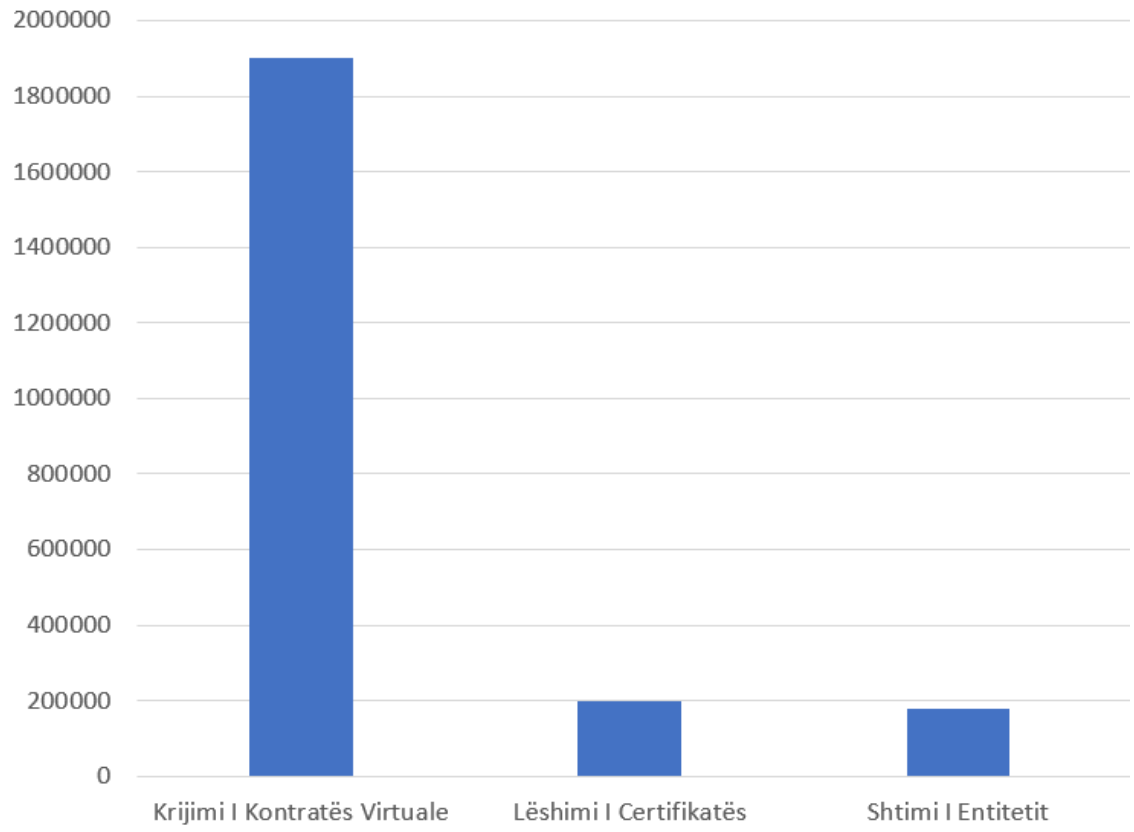


6.1 - Grafiku i konfirmimit të transaksionit gjatë 2021 (Në sekonda) [18]

Grafiku në figurën 6.1 tregon kohën që i duhet blloqeve të përfshihen në rrjet. Kjo kohë matet në sekonda, dhe varion jo më shumë se 15 sekonda.

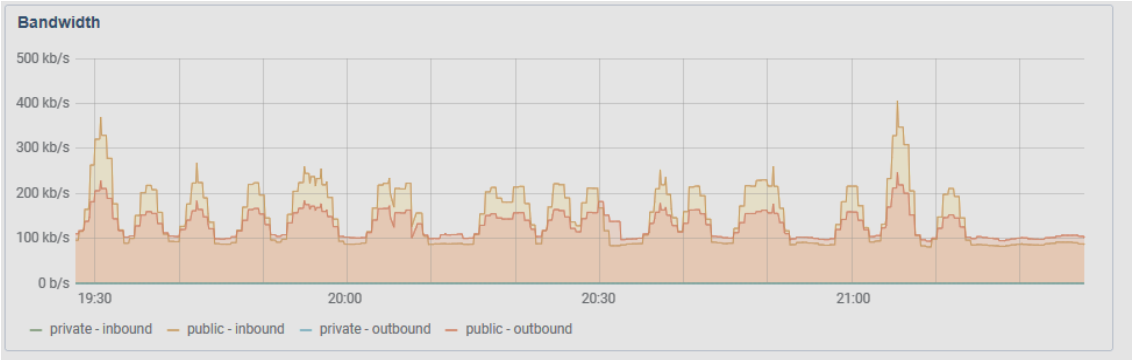


6.2 - Grafiku i cmimit të gasit që nga 2020 (Në GWei) [18]

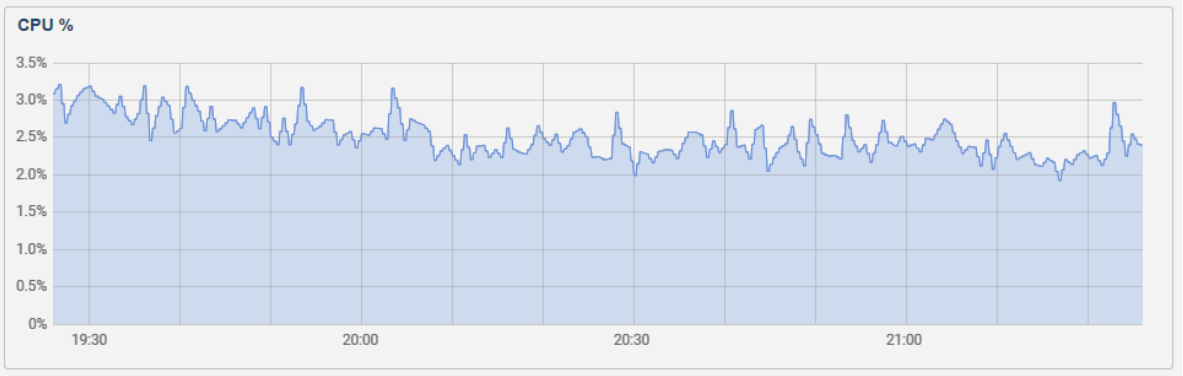


6.3 - Kostoja për cdo veprim mbi kontratën virtuale (në gas)

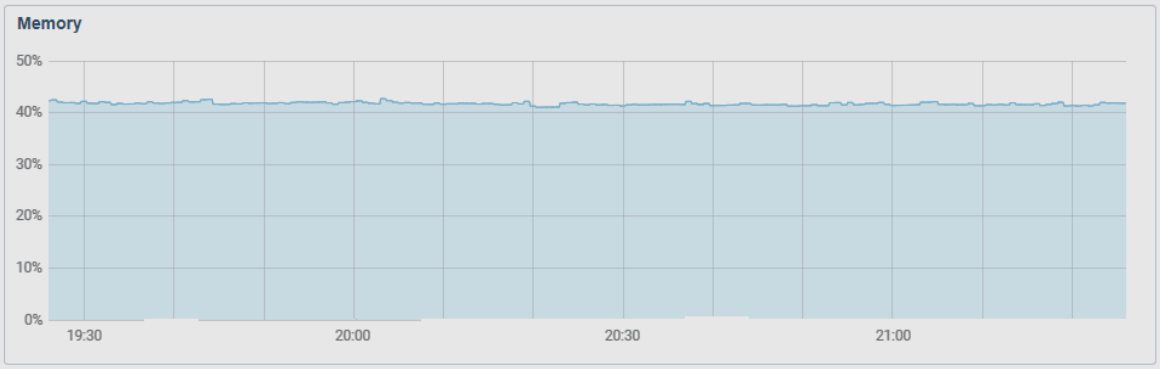
Më sipër në figurën 6.3 tregohet vizualisht gasi që merr cdo veprim. Krijimi i kontratës virtuale dhe shtimi i entitetit janë vlera kryesisht statike, ndërsa lëshimi i certifikatës luhatet në varësi të madhësisë së të dhënave. Në momentin e matjes u hodhën përafërsisht 200 bajt të dhënash.



6.4 - Përdorimi i bandwidth



6.5 - Përdorimi i CPU



6.6 - Përdorimi i RAM

Për grafikët e mësipërm është përdorur DO-Agent, një mjet që vjen bashkë me serverat e DigitalOcean. Gjithashtu cdo server ka 2GB RAM, 1 Core dhe lidhje 100Mbit/s dhe grafikët janë matur për një periudhë rreth 2 orë. Vihet re që kryesisht resurset janë stabël, dhe pjesa e ndërveprimit me kontratën shihet tek rritja e trafikut të networkut.

KAPITULLI 7: KONKLUZIONE

Në përfundim të këtij projekti kemi një sistem funksional të lëshuar në blockchain. Ky projekt përfshin lëshimin e disa certifikatave të pronësisë të plotësuara me të dhënat sipas standarteve të paracaktuara, si dhe verifikimin sa më të lehtë nga përdoruesi. Përdoruesit i kërkohet të plotësojë vetëm dy fusha, ID e pronësisë dhe celësi i dekriptimit. Proceset kanë një sinkronizim midis njëri-tjetrit duke mundësuar përdorimin e sistemit në cdo kohë nga cdo paisje me akses interneti. Në cdo rast zgjedhje, kemi krijimin e një blockchain privat, në mënyrë që të bëhet lehtësisht zgjedhja nga entiteti përdorues. Kemi marrë parasysh që kompleksiteti i implementimit nuk duhet të kalojë tej mase implementimet aktuale, dhe kostoja e mirëmbajtjes së sistemit duhet të jetë pothuajse aq sa e sistemit aktual. Kjo vihet re nga serverat me specifikime të ulëta që u përdorën në këtë shembull. Nga kjo punë u shfaq qartë procesi i lëshimit dhe verifikimit të certifikatave.

7.1. Zhvillime të mëtejshme

Ndërtimi i këtij projekti tregoi se koncepti është më se i realizueshëm. Mungojnë optimizime dhe verifikime shtesë në blloqet bazë të sistemit, por këto do të vihen re vetëm me përdorim të gjerë të projektit. Mbetet për tu ndryshuar dhe përshtatur pamja grafike dhe ndërfaqet vizuale të përdoruesit.

Një nga pikat e para që duhet të adresohet më tej është studimi i platformave të ndryshme. Mënyra e strukturimit të këtij projekti mundëson kalimin lehtësisht në platforma të tjera, dhe si rrjedhojë në rast se gjenden platforma të tjera më eficiente, ato duhet të merren në konsideratë.

Punë të mëtejshme përfshijnë metodologjinë që do të përdoret për paisjen e përdoruesit apo pronarit me celësin e enkriptimit. Këto mund të variojnë nga komunikimi drejtpërdrejt, deri tek dërgimi me e-mail, numër telefoni apo me anë të një kodi QR. Në rastin e këtij të fundit përdoruesi duhet të ketë mundësinë që ta skanojë kodin dhe ti shfaqet certifikata direkt në ekran duke mos pasur nevojë për veprime ekstra.

Megjithëse kemi përdorur teknologji si PWA për të ulur kohën e zhvillimit, në rast se do të kishim burimet e mjaftueshme do preferoheshin aplikacione enkas për Android, iOS dhe Windows. Kjo pasi aplikacionet native janë më të shpejta dhe rrisin ndjeshëm performancën. Ndryshe nga PWA të cilat punojnë mbi një browser ku ekzekutohet kodi i JavaScript, dhe renderimi i elementëve kërkon më shumë burime.

REFERENCA

[1] Nicole Radziwill (Reviewed by) (2018), “Blockchain Revolution: How the Technology Behind Bicoin is Changing Money, Business, and the World,”
[2] Muhammad Umer Shabbir (2021), “Blockchain in Real Estate Sector: Benefits and Challenges”
[3] Sharyar Wani, Mohammed Imthiyas (2021), Hamad Almohamedh, Khalid M Alhamed, Sultan Almotairi, “Distributed Denial of Service (DDoS) Mitigation Using Blockchain - A Comprehensive Insight”
[4] Satoshi Nakamoto (2008), “Bitcoin: A Peer-to-Peer Electronic Cash System”
[5] Soujanya Ponnapalli, Aashaka Shah, Amy Tai, Souvik Banerjee, Dahlia Malkhi, Vijay Chidambaram, Michael Wei (2020), “Rainblock: Faster Transaction Processing in Public Blockchains”
[6] Divya Guru, Supraja Perumal, Vijayakumar Varadarajan (2021) - “Approaches towards Blockchain Innovation: A Survey and Future Directions”
[7] Christopher Natoli, Jiangshan Yu, Vincent Gramoli, Paulo Veríssimo (2019), “Deconstructing Blockchains: A Comprehensive Survey on Consensus, Membership and Structure”
[8] Gokhan Sagirlar, Elena Ferrari (2018), “Hybrid-IoT: Hybrid Blockchain Architecture for Internet of Things - PoW Sub-blockchains”
[9] Vitalik Buterin (2013), “A next generation smart contract & decentralized application platform”
[10] Andreas Biørn-Hansen, Tim A. Majchrzak, Tor-Morten Grønli (2017), “Progressive Web Apps: The Possible Web-native Unifier for Mobile Development”
[11] Juan Benet (2014), “IPFS - Content Addressed, Versioned, P2P File System”
[12] Jürgen Cito, Vincenzo Ferme, University of Lugano, Harald C. Gall (2016), “Using Docker Containers to Improve Reproducibility in Software and Web Engineering Research”
[13] Roman Matzutt, Jan Pennekamp, Erik Buchholz, Klaus Wehrle (2020), “Utilizing Public Blockchains for the Sybil-Resistant Bootstrapping of Distributed Anonymity Services”
[14] André Glória, Francisco Cercas (2017), “Design and implementation of an IoT gateway to create smart environments”
[15] A. Manoj Athreya, Ashwin A. Kumar, S. M. Nagarajath, Gururaj H L (2021), “Peer-to-Peer Distributed Storage Using InterPlanetary File System”
[16] *React A JavaScript library for building user interfaces* (Aksesuar Shtator 2021) <https://reactjs.org>
[17] Joan Daemen, Vincent Rijmen (2003), “AES Proposal: Rijndael”
[18] The Ethereum Blockchain Explorer (Aksesuar Maj 2021) <https://etherscan.io>