Academia de Studii Economice din Bucureşti
Facultatea de Cibernetică, Statistică şi Informatică Economică
Specializarea Informatică Economică

Platformă Speculativă de cripto-active generate ad hoc

Lucrare de Licenţă

Coordonator Ştiinţific:
dr. Vinte Claudiu

Absolvent:
Petroschi Matei

Bucureşti

Cuprins

Abstract

Lucrarea curentă descrie intreg procesul de funcționare al unei platforme de tranzacționare de cripto-active valorificate printr-un sistem blockchain folosit de o rețea distribuită de clienți conectați direct între ei. Lucrarea încearcă să demonstreze cum valoarea reală și cuantificabilă poate fi atribuită oricărei bucăți specifice de informație arbitrară indiferent de natura ei și că acea valoare este transferabilă între entități corelată perfect cu conținutul transferat.

Un obiectiv secundar este cel de arăta procesul de dezvoltare a unui sistem descentralizat în paradigma *Web 3.0* autosuficient care este intrinsec rezistent cenzurii si entropiei generale prin independența față de o entitate centrală oarecare care controlează și facilitează sistemul.

Introducere

Menirea platformei este să permită oricărui participant să încarce conținut informatic arbitrar (e.g. imagini, videouri, clipuri audio, text etc.) într-o rețea descentralizată în care aceste bucăți de conținut informatic să poată să aibă valoare recunoscută de ceilalți participanți.

În momentul încărcării utilizatorii vor putea să declare două lucruri fundamentale despre activul pe care îl creează:

- Numarul de unități finite de activ care există și pot fi tranzacționate
- Valoarea inițială per unitate exprimată în *cripto-monedă* (LSK)

Rezultatul final este un activ împărțit într-un număr fixat de copii unice care acum se află în portofoliul utilizatorului. Acesta poate mai departe să creeze oferte de vănzare pentru câteva sau toate instanțele din activul respectiv, iar participanții la piață vor decide dacă valoare atribuită inițial conținutului este corelată precis și în caz pozitiv vor alege să cumpere activul respectiv.

În cel mai practic sens acest activ generat de către utilizator devine în final un *Store of Value* [7] transferabil în interiorul sistemului.

Un obiectiv secundar al platformei în sine este și transferul de proprietate intelectuală de la un participant la altul cu aplicabilitate legală și în exteriorul sistemului în condițiile în care proprietarul inițial în interiorul platformei este proprietarul sau creatorul original al conținutului și în exteriorul platformei.

O Scurtă istorie și stadiul cunoașterii

Contextul tehnologic

Evolutia tehnologiei relevante contextului modern în care toate unitățile computaționale sunt interconectate este marcată de trei paradigme importante: Web 1.0, Web 2.0 și Web 3.0.

Web 1.0

Web 1.0 se refera la primul stadiu al evoluției *World Wide Web*. În vremurile astea erau puțini creatori cu o majoritate de utilizatori care consumau conținut. Pagini Web personale erau comune și erau reprezentate în principal de pagini statice hostate pe webservere facilitate direct de către ISP-uri (Internet Service Provider), prin alte servicii de hostare web sau chiar direct pe o mașină deținută de creator.

Un fel de a vedea web-ul în timpul acela era ca un simplu CDN (Content Delivery Network) care permitea utilizatorilor să consume o bucată de conținut și atăt.

Patru elemente de design esențial al unui site Web 1.0 includ:

- Pagini statice
- Conținut servit direct de pe *file system*-ul serverului
- Pagini construite folosind *SSI* [11] sau *CGI* [4]
- Tabelele și Ramele erau folosite pentru poziționarea și aliniarea elementelor în pagină [12]

Modelele de monetizare folosite în vremea asta, dacă există, sunt primitive și specifice fiecăruia, nu există servicii comune de monetizare sofisticate.

Web 2.0

Web 2.0 e paradigma care a dat naştere internetului şi aplicaţiilor în felul în care le cunoaştem în ziua de astăzi. O poreclă pentru *Web 2.0* este *The participative social web*. Această perioadă e caracteristică facilitării interacţiunii între utilizatori prin platforme sociale robuste care permit persistenţa acţiunilor, dar mai important decât asta este normalizarea interacţiunilor intre servicii şi standardizarea consumului şi controlului datelor. [12]

În această perioadă se popularizează conceptul de *Application Programming Interface* (API) care prin *Data Interchange Formats* comune pot deschide accesul oricui la izvoare aproape nelimitate de date brute care pot fi manipulate în varii scopuri nepredeterminate.

Browserele Web evoluează să permită utilizarea de metodologii precum *Asynchornous JavaScript and XML* (AJAX) [1] care permit unei pagini web să ceară date de la servicii externe fără să blocheze întreg procesul şi şă aştepte rezultatul asincron, lucru care determină dezvoltarea *Framework*-urilor complexe JavaScript şi în final dezvoltării de aplicaţii web dinamice care nu mai sunt dependente de server pentru generarea paginilor. Se trece de la *Multi Page Applications* la *Single Page Applications* (SPA) unde clientul încarcă conţinutul static şi codul sursă al aplicaţiei, iar conţinutul dinamic este cerut doar în momentul în care este consumat.

Cinci caracteristici tehnologice importante ale *Web 2.0* sunt:

- Organizarea, clasificarea şi consumarea liberă a informaţiei în manieră colectivă
- Conţinut dinamic
- Utilizarea de API-uri externe
- Applicaţii *responsive* de *social media*
- Interactivitatea

Pe lângă dezvoltările tehnologice ale vremii la fel de importante au fost dezvoltările în modelele de business şi vectorii de monetizare. Putem în sfărşit să vedem o serie de modele de monetizare standardizate şi servicii externe care facilitează procesul de monetizare:

- Targeted Advertising via Google Ads (aka AdWords)
- Crowdfunding via servicii precum Kickstarter sau Indigogo pentru generarea de fonduri iniţiale şi Patreon pentru donări recurente direct către firme şi creatori
- Plată directă prin transfer bancar sau printr-un serviciu ca PayPal sau Stripe cu modele de business precum (Pay2Play sau Free2Play) pentru accesarea unui bun

sau serviciu sau pentru efectuarea tranzacţiilor între terţe părţi pe platforme cu această menire

- *Data Mining*-ul devine prolific ca manieră de a monetiza activitatea utilizatorilor prin tracking efectuat la nivel meta între toate site-urile şi aplicaţiile web folosite de utilizatori; Site-urile participă voluntar în piaţa de date brute punând *cookie*-uri care permit identificarea utilizatorilor şî raportarea către Brokeri de Date care remunerează datele culese
- Brokerii de Date agregă date de la toate site-urile şi aplicaţiile web care participă în procesul de *data mining* pentru a le vinde în variantă brută sau prelucrată deja în profile psihologice mai departe în principal, dar non-exclusiv, la firme de marketing care îşi adaptează strategiile şi la servicii precum Google Ads pentru a arăta reclame de interes pentru fiecare utilizator în parte

Toate acestea au dus la sporirea dezvoltării de aplicaţii web şi site-uri într-o lume post *Dot-com bubble* şi la resurgenţa creatorilor independenţi care ajung să fie în sine motorul economiei Web.

Web 3.0

Web 3.0 este cel mai recent val tehnologic care încă este în procesul de definire. Aflându-ne încă în mijlocul evenimentului elementele caracteristice sunt încă în discuţie, nu o să avem un răspuns clar a ceea ce va reprezenta noua paradigmă pană în momentul cristalizării. Există multă suprapunere cu penultimul val ai cărui caracteristici apar parţial şi în definiţiile celui curent.

Dat fiind că multe dintre elementele *Web 2.0* vor fi rădăcinile pentru multiplele mutaţii în diferitele directii pe care le va lua acest val acest lucru nu e surprinzător, astfel încât diferenţierea se va face pe de-o parte la nivel de nuanţă în abordare şi implicaţii sociale, politice şi economice.

O serie de caracterstici populare sunt:

- *Semantic Web*, ideea de dezvoltare a soluţiilor într-o manieră care permite interacţiunea la nivel semantic cu informaţia
- Prolifierea Inteligenţei Artificiale în sistemele care croiesc colecţia de conţinut pe care o consumă utilizatorul în sine folosindu-se de *folksnonomia* creeată prin utilizarea organică a aplicaţiilor chiar de către utilizatori
- WebAssembly
- Omniprezenţa datelor şi a tehnologiilor Web în toate dispozitivele

> “I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the contenţ linkş and transactions between people and computers. A “Semantic Web”, which makes this possible, has yet to emerge, but when it doeş the day-to-day mechanisms of trade, bureaucracy, and our daily lives will be handled by machines talking to machines. The “intelligent agents” people have touted for ages will finally materialize.’ ’
>
> — Sir Tim Berners-Lee

Dar pe lângă cele menționate mai sus este discutabil faptul că cea mai importantă caracteristică a valului *Web 3.0* și cea care va fi amintită de istorie în viitor ca fiind determinantă pentru această perioadă va fi prolifierea conceptului de *Descentralized Application* (dApp), mișcarea către a dezvolta aplicații într-o arhitectură descentralizată bazată pe implementări *Blockchain*.

Crypto-monedele au reprezentat prima implementare a tehnologiei *Blockchain* care au popularizat conceptul și au dovedit că infrastructura modernă poate în sfârșit să-l suporte, în schimb lucrurile nu se termină aici.

Blockchain este o formă de bază de date distribuită și descentralizată care printr-un mecanism de rezoluție a consensului între agenții participanți poate să garanteze intrinsec identitatea, validitatea și imutabilitatea operațiunilor pe care le înscrie și replică pe toate dispozitivele participante.

Un *dApp* este un tip de applicație care la suprafață funcționează la fel ca oricare alta, dar nu este dependentă de niciun server central, conexiunile dintre client și rețea fiind de tip Peer-to-Peer, conținutul persistent dinamic fiind reținut în blockchain-ul comun tuturor nodurilor din rețea.

La finalul erei *Web 2.0* deja deveneau evidente anumite probleme cu felul în care se face business-ul pe web. Principalele fiind:

- Consolidarea tuturor resurselor informatice într-un număr foarte redus de servicii monolit deținute de agenți economici cu prea multă putere
- Lipsa de autoritate asupra informației publicate pe servicii în sensul in care deși proprietatea intelectuală a unui creator îi aparține, orice firmă are puterea de a controla dacă acea proprietate este accesibilă sau nu pe platforma respectivă
- Cenzura selectivă după discreția firmelor și manipularea trendurilor
- Colectarea clandestină de date fără remunerarea utilizatorilor

În ultimii 3-4 ani eforturile oamenilor concentrați pe problemele menționate mai sus s-au manifestat în direcția eliminării complete la nivel arhitectural a oricărei entități centrale care să poată să controleze conținutul, dând înapoi puterea cibernetică utilizatorilor care acum vor avea să fie atât clientul cât și serviciul în sine.

O altă problemă evidentă a lumii vechi este dependența față de *advertising* ca vector financiar principal. Când au început să devină populare soluții precum AdBlock, apăruseră atât în rândul creatorilor cât și al firmelor frici legate de sustenabilitatea modelului economic web și dacă procesul poate să continue.

Golul lăsat de *ads*-urile în minus a fost din fericire umplut de soluțiile de crowdfunding care au apărut, lucru care a fost benefic din multe puncte de vedere, nu cel din urmă fiind faptul că acum tabla de joc începe să dea semne de normalizare a distribuției de valoare asupra conținutului informatic creeat doar prin simplul fapt că acum oamenii pot să aleagă voluntar pe cine să plătească după gust lucru care a creeat o formă de clasă de mijloc între creatorii de pe internet.

Web 3.0 construiește mai departe pe conceptul de crowdfunding și vine și cu soluții noi. Brave este un browser lansat în 2019 care include nativ toate facilitățile necesare pentru funcționarea dApp-urilor, el în sine fiind un dApp cu propriul lui sistem *Blockchain* cu cripto-monedă. Brave va permite tuturor dApp-urilor care se integrează cu standardul prezent să primească bani direct de la utilizatorii care vor să plătească cu crypto-moneda brave într-un mod foarte intuitiv direct din browser.

Când vine vorba de locul *advertising*-ului în lumea *Web 3.0*, având în vedere că deja browser-ele încep să blocheze *ads*-urile implicit este destul de clar că nu vor mai fi la fel de intrusive și lumea va continua trendul către alte modele de monetizare. Chiar și așa ele nu vor dispărea complet, o altă funcționalitate a browser-ului Brave este vizualizarea voluntară remunerată a *ads*-urilor, modelul de *advertising* devenind mai sofisticat prin faptul că acum utilizatorul trebuie luat în calculul marginilor financiare împarțite pe lângă serviciul care ține *ads*-urile în discuție și evident *advertiserul* care deține *ads*-urile și contractele cu firmele ale căror produse și servicii sunt prezentate. Utilizatorii finali vor trebui insentivați direct să consume ca să participe voluntar în orice mecanism de monetizare, duse sunt zilele sprijinului involuntar.

Acesta este contextul tehnologic în care a fost dezvoltată aplicația prezentată mai departe în această lucrare.

Contextul politic

Dat fiind gradul de consolidare mediatică din prezent devine din ce în ce mai evident că infrastructura curentă este prea vulnerabilă controlului de către firmele care deţin platformele folosite de populaţie şi de către actori statali care influenţează direcţia discursului politic digital prin mecanisme externe platformelor în sine. În alte cuvinte este aparent faptul că fluxul informatic este prea uşor influenţat de legile şi ordonanţele statelor lumii şi a moderării draconice efectuate de firme pentru a limita selectiv accesul la informaţie.

Lucrarea de faţă invocă anumite presupoziţii filosofice în sensul ăsta:

1. Într-o lume *post-adevăr* gradul de entropie informaţională trebuie mărit nu micşorat
2. Heterogenitatea naturii entităţilor care pot să impacteze semnificativ fluxul informaţional trebuie maximizată
3. Orice agent economic sau actor statal trebuie să subscrie aceluiaş set de reguli ca oricare alt subset de indivizi care participă colectiv la producerea conţinutului, altfel spus *software*-ul trebuie construit în aşa fel încât să nu poată fi controlat prin alte maniere în afară de propriile lui mecanisme organice interne

În momentul scrierii lucrării, în mijlocul pandemiei COVID-19, Congresul american dezbate *Eliminating Abusive and Rampant Neglect of Interactive Technologies ("EARN IT") Acţ* care urmăreşte să încarce firmele private cu mai multă responsabilitate legală de a preveni utilizatorii din a încărca material ilegal, în specific legat de prostituţie. Legea nu face distincţie între canale publice şi private în interiorul unei aplicaţii având ca potenţial rezultat forţarea firmelor să creeze *backdoor*-uri sau să renunţe complet la sistemele lor de end-to-end encryption pe care le folosesc utilizatorii lor pentru a putea să monitorizeze activitatea în întregime pentru a o modera. [3]

Combinaţia cu *European Union Directive on Copyright in the Digital Single Market* care includea infamul *Articol 13* care responsabilizează legal firmele să prevină încărcarea ilegală de proprietate intelectuală face ca firmele să fie obligate să implementeze măsuri draconice de control informatic pe lângă cele pe care chiar ele ar fi vrut să le implementeze, lucru care are ca potenţial rezultat un grad de sterilitate informaţională contraproductiv presupunerilor menţionate mai sus. [2]

Scopul acestei secțiuni nu este menit activismului politic, în schimb este de a prezenta o secțiune verticală a motorului ideologic care conduce eforturile de reconstrucție infrastructurală și arhitecturală având ca unul dintre obiectivele centrale libertatea și autodeterminarea informatică.

Soluții existente

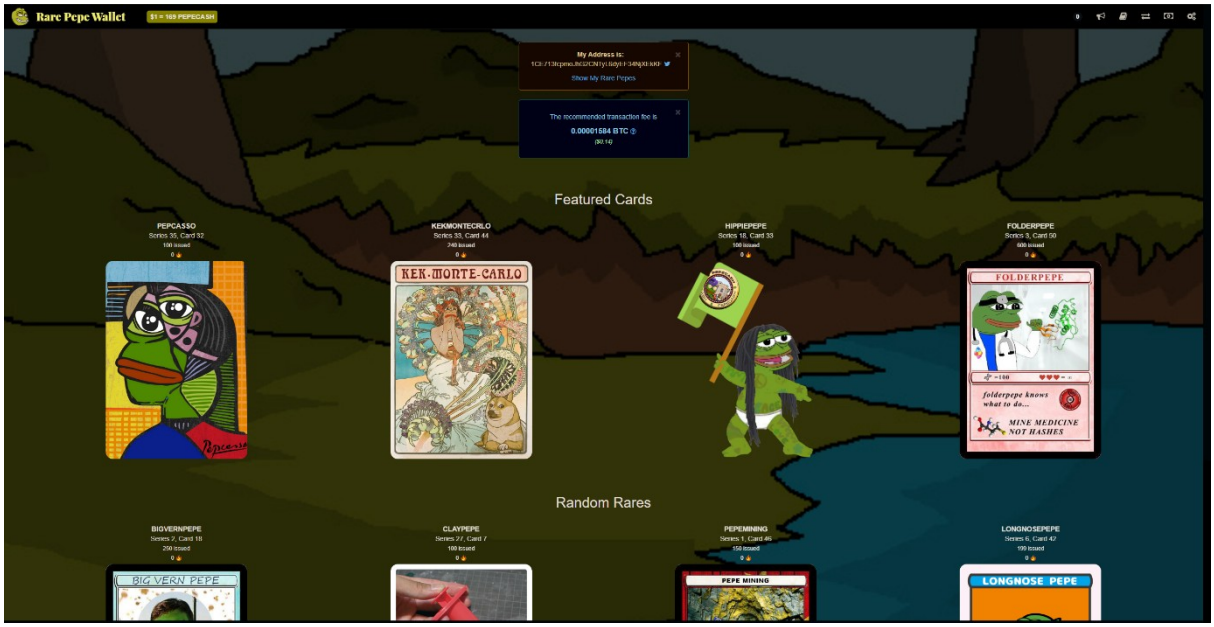

Figura 1: RarePepeWallet (rarepepewallet.com, 2020)

Rare Pepe Wallet este sistemul care a inspirat inițial această lucrare. Construit folosind un sistem dApp bazat pe Bitcoin, este o aplicație simplă care permite utilizatorilor să cumpere și să vândă instanțe de “Rare Pepes” a căror prezență și cantitate este controlată central. [13]

Listarea este precurată de către creatorii platformei din propuneri făcute de oricine. Piața este încă în folosință activă, iar oamenii continuă să facă speculă pe informație digitală oarecare în forma instanțelor artistice având ca subiect comun personajul folkoric tradițional al internetului Pepe.

Aici am întâlnit pentru prima dată această idee de cuantificare a valorii reale exprimată în monedă efectivă a unei bucăți de informație arbitrară folosind un mecanism de piață liberă în care agenții participanți decid organic valorile fiecăreia.

Implicațiile potențiale sunt mai largi decât cele artistice. Lucrarea curentă încearcă să prezinte cum dacă funcționalitățile aplicației ce înconjoară piața sunt extinse în de ajuns, rezultatul este un mecanism de valorificare memetică completă. Postări de orice fel pot fi schimbate în aceași manieră. Text, video, audio, idei, concepte, ideologogii, orice structură abstractă indiferent de gradul de complexitate poate reprezenta subiectul speculei.

Propunere Arhitecturală a Soluției Software

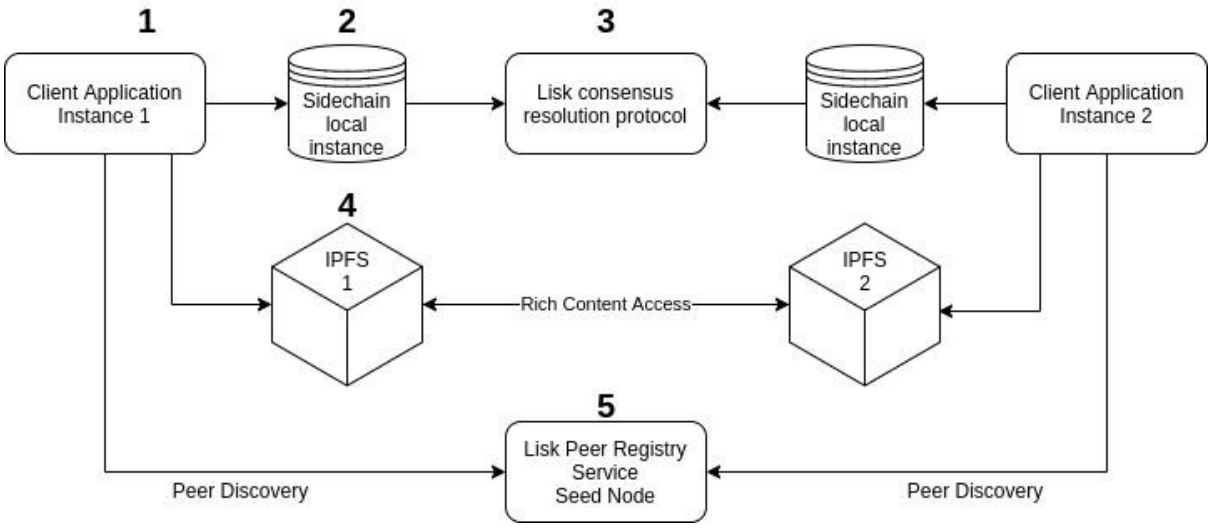

Figura 2: Arhitectura Generală

1. Applicație Angular 2+ folosită pentru interacțiunea la nivel de peer cu rețeaua sistemului, folosită de utilizatorul final pentru a creea, vinde și cumpăra crypto-active
2. Instața locală a *sidechain*-ului Lisk în care se înregistrează informațiile despre crypto-active, date despre utilizator
3. Protocolul de consens este procesul prin care toate adăugirile la blockchain sunt agregate, validate și propagate înapoi în rețea ca noua versiune de blockchain comuna tuturor nodurilor
4. Instața de IPFS (InterplanetaryFileSystem) prin care se stochează local conținutul bogat corelat cu crypto-activul creeat de utilizator și îl expune la internet
5. Componenta din Lisk care deservește drept *Seed Peer* și înregistrează nodurile conectate (Singura componentă centrală din arhitectură)

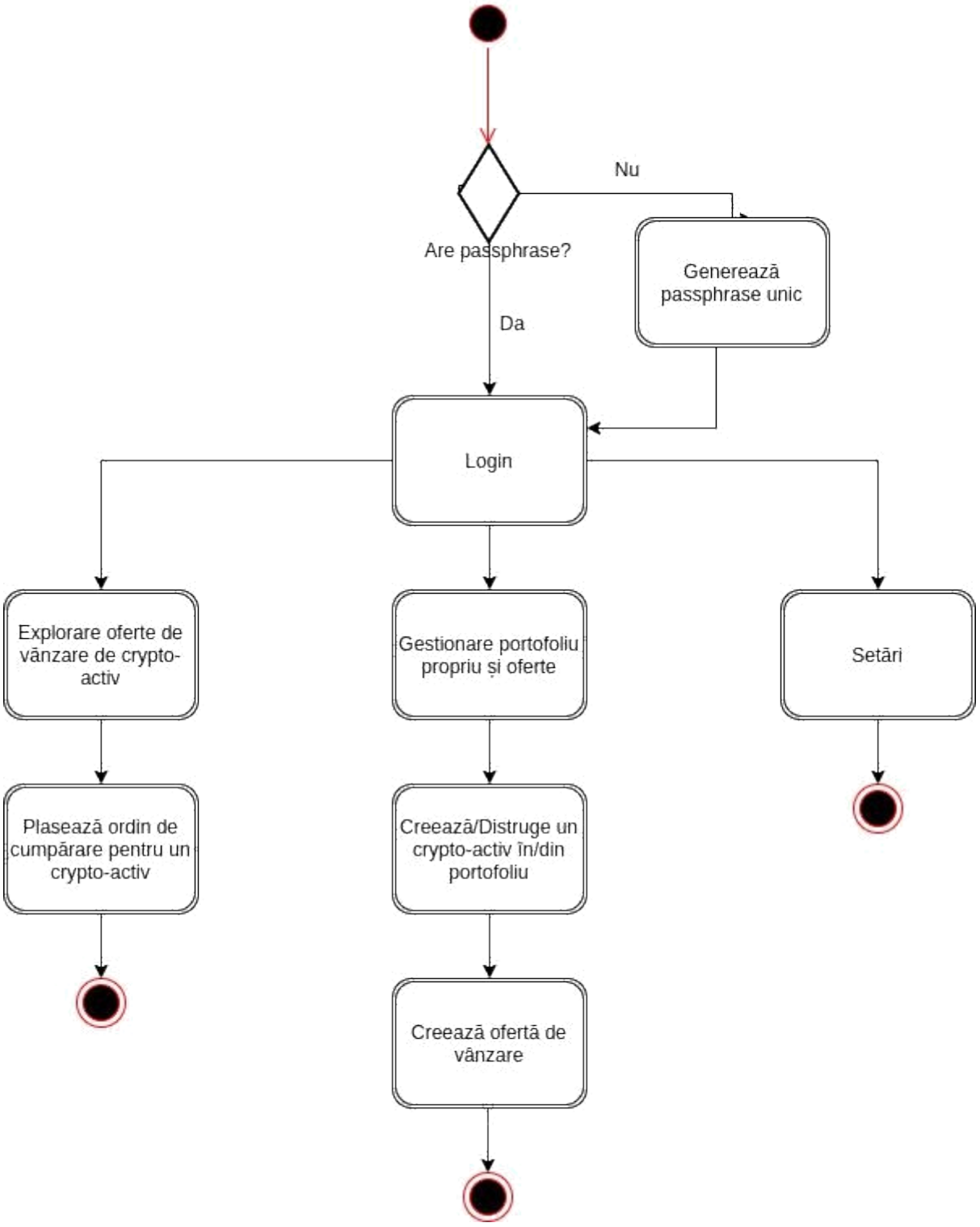

Figura 3: Diagrama de utilizare a aplicației client

Descrierea funcţiilor principale

Login

În comparație cu aplicațiile normale, acest tip de aplicație nu necesită un proces separat de înregistrare, în momentul primei utilizări, utilizatorul poate să genereze pe pagina de login *passphrase*-ul unic cu care se va identifica de acum în colo. Exemplu de *passphrase*:

horn future wife theme inherit ice buffalo drop chimney forum boss educate

Acest *passphrase* este spart de modulul *@liskhq/lisk-cryptography* în perechea de chei unice asociate utilizatorului care sunt folosite pentru autorizarea tuturor acțiunilor legate de utilizatorul respectiv.

Nimic altceva nu este necesar în afară de acest *passphrase* care trebuie ținut minte de utilizator și nu va putea niciodată fi schimbat fără a pierde întreagă activitatea din sistem.

În cel mai practic sens ‘contul’ utilizatorului este creeat de abia în momentul în care se înregistrează prima tranzacție cu adresa și cheia publică expusă în blockchain fie că există date asociate sau e o transacție nulă.

Creearea de crypto-active

Pe pagina cu portofoliul utilizatorului acesta este capabil să creeze un crypto-activ cu printr-un modal în care poate să introducă datele relevante și să încarce conținutul bogat asociat activului respectiv.

Conţinutul bogat poate să fie de orice natură (text, photo, video, audio etc.) și în momentul încărcării pe platformă este de fapt înregistrat pe instața locală de IPFS pentru a fi expus internetului. În momentul în care cineva vizualizează crypto-activul de pe alt dispozitiv acela este de abia momentul când conținutul este citit de pe dispozitivul origine și *cache*-uit pe dispozitivul consumator.

Vânzarea şi cumpărarea de cripto-active

Principala activitate realizată pe platformă este cea de vânzare și cumpărare de cripto-active generate de utilizator. Acest lucru se realizează descentralizat prin intermediul unui cont virtual terță parte care înregistrează toate tranzacțiile de pe platformă. Rezultatul final este o logică transacțională de tip *escrow* pentru a asigura securitatea tranzacțiilor între participanții la piață prin care cripto-activul este mutat în posesia contului terț până în momentul în care un participant este capabil să satisfacă condițiile tranzacției.

Tehnologii utilizate

Dată fiind amploarea proiectului colecția de tehnologii utilizate pentru atingerea fiecărui obiectiv este relativ largă:

Angular 2+

Framework-ul de frontend utilizat pentru realizarea componentei interactive a proiectului, Angular 2+ este un framework SPA (Single Page Application) care funcționează pe bază de componentă.

Componentele în Angular sunt făcute în așa fel încât să fie bucăți modulare care conțin atât elementele necesare pentru stilizare cât și codul sursă care va fi să fie executat în relație cu componenta respectivă și componentele înconjurătoare. Ele pot fi creeate într-o formă cât de atomică sau complexă este nevoie astfel încât să poată fi folosite ca orice tip de element de UI autoconținut în interiorul unui șablon HTML dinamic.

Angular 2+ este gândit în jurul unui pattern arhitectural la nivel de aplicație cunoscut în accepțiunea comună curentă sub denumirea de MVW (Model-View-Whatever, Where Whatever stands for “whatever works for you”). În perioada de tranziție între Angular.js șî Angular 2+ era pusă în discuție de către comunitate clasificarea lui Angular ca framework într-una dintre *pattern*-urile următoare:

- MVC (Model-View-Controller)
- MVVM (Model-View-ViewModel)
- MVP (Model-View-Prezenter)

În final dat fiind faptul că important era partea comună între aceste *pattern*-uri și anume ideea de decuplare a *View*-ului (Partea structurală și stilistică), Model (Structura de date arbitrară asociată *View*-ului) și logica efectivă, eticheta de MVW a devenit un descriptor popular. Folosirea componentelor reutilizabile făcute accesibile prin sistemul de dependency injection în combinație cu o multitudine de feluri în care se poate obține *data-binding* între elementele din *View* Și model determină o practică mai scalabilă în comparație cu oricare dintre *pattern*-urile vechi privite izolat.

Angular 2+ vine cu o suită de abstracții fundamentale care fluidizează procesul de dezvoltare dramatic, printre ele numărându-se și sistemul robust de routing și HttpClient-ul puse la dispoziție.

NestJS

NestJS este un framework de *backend* care a fost dezvoltat de la început să fie foarte similar cu Angular dată fiind popularitatea percepută și scalabilitatea efectivă a *pattern*-ului arhitectural implementat.

Aceleași principii de design din Angular se aplică și în NestJS lucru care facilitează un proces de dezvoltare în oglindă între porțiunea de *frontend* și *backend*, iar sistemul de *dependency injection* și *design pattern*-urile native precum *Interceptor*, *Guard* și *Factory* au ajuns să fie standard și în lumea *backend*-ului bazat pe Node.js.

NestJS folosește Express ca motor de abstracție HTTP implicit dar poate fi înlocuit cu Fastify.

Ambele *framework*-uri folosesc TypeScript ca limbaj de programare efectiv.

Node.js

Node.js este un *runtime environment* de JavaScript care folosește motorul V8 original creeat și menținut de Google și implementează standardul ECMA.

TypeScript

Este un superset de JavaScript care este transpilat de *TypeScript Compiler* în orice versiune de JavaScript și poate să fie rulat în aceleași condiții.
TypeScript a fost creeat pentru a ameliora problemele care au apărut în evolutia limbajului pe care se bazează în lunga lui istorie plină de evenimente care au cerut noi funcționalități pentru care nu fusese inițial menit.

Express

Este un *framework* de abstracție HTTP bazat pe sistemtul de server și socket nativ Node.js.

Lisk

Lisk este un sistem de blockchain construit pe Node.js care manifestă o funcționare foarte performantă în comparație cu majoritatea soluțiilor de pe piață și a fost construit de încă de la început cu obiectivul de a servi drept o platformă largă pe care să se integreze *dApp*-uri care să se folosească de *Mainnet* pentru a vira valoare reală în interiorul rețelei lor private respective și ca framwork de *sidechain*-uri cu propria logică de proprietate.

În comparație cu Ethereum, cel mai mare sistem *blockchain* care e menit dezvoltării *dApp*-urilor, Lisk este construit în întregime folosind JavaScript executat pe Node.js și își concentrează filosofia de dezvoltare pe conceptul de tranzacții personalizate în pofida unui limbaj propriu precum Solidity și a conceptului de *smart contract* și folosește PostgreSQL ca bază de date pentru menținerea persistenței locale a *blockchain*-ului.

Această configurație garantează un prag de cunoaștere necesar pentru lucrul cu sistemul Lisk foarte scăzut și permite un volum potențial mai mare de ingineri care pot să înceapă imediat să construiască *dApp*-uri fără a pierde timpul învățând un limbaj nou de programare sau un sistem de gestionare ale bazelor de date nou. Proiectul Lisk a putut să ajungă în starea în care este astăzi de stabilitate atât la nivel tehnic cât și la nivel de securitate financiară pentru că încă de la început a putut să se folosească de inerția și bogăția tehnologică a ecosistemului Node.js, fapt ce a determinat o evolutie foarte agresivă într-un timp foarte scurt.

Sidechains

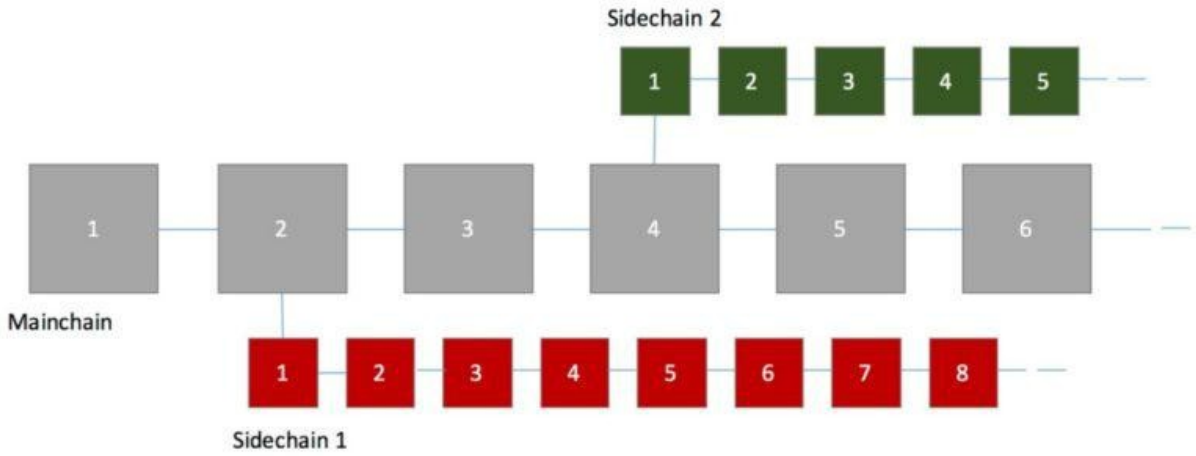

Figura 4: Sidechains (Diagoal, 2018)

Lisk permite dezvoltarea de *dApp*-uri independente care pot să efectueze operațiuni arbitrare personalizate după discreția inginerilor participanți folosind ideea de *Sidechain*.

Un *sidechain* este o instană independentă de blockchain compatibilă la nivel fundamental cu *mainchain*-ul Lisk. Faptul că *dApp*-urile folosesc *sidechain*-uri proprii pentru operațiunile care necesită logică proprie are mai multe avantaje în comparație cu alte sisteme ale căror aplicații conectate trebuie toate să utilizeze același *blockchain* comun în aceeași rețea printre care și:

- Abilitatea de a creea tranzacții personalizate care nu trebuie să respecte standarde predefinite comun de comunitate
- *Mainnet*-ul fiind complet separat nu este încărcat suplimentar nici la nivel de rețea și nici la nivel de informație irelevantă pentru majoritatea participanților care trebuie stocată
- Inginerii pot să vină cu soluții ineficiente fără să afecteze negativ performanța sistemului întreg

Delegatated Proof of Stake

Delegated Proof of Stake sau DPoS este protocolul de rezoluție a consensului folosit de proiectul Lisk și este lucrul care a permis atingerea performanței ridicate manifestată în prezent.

Metrica principală de performanță pentru sistemele *blockchain* este *block time*, timpul necesar pentru ca un set de tranzacții să fie agregate și validate de protocolul de consens pentru a fi înscrise ca un nou *block* în *blockchain*.

Pentru comparație:

- Bitcoin block time: ~10min [8]
- Ethereum block time: ~20s [8]
- Lisk block time: ~10s

Majoritatea sistemelor de până acum foloseau mecanisme de consens de tip *Proof of Work* sau PoW, care necesită rezolvarea unei probleme cryptografice complexe pentru a da dreptul unui nod din rețea să înscrie un *block* nou în blockchain, acest lucru înseamnă că un mecanism de acest tip va necesita mereu putere computațională relativă foarte ridicată cu un consum real de energie electrică corelat și deci cu costuri foarte mari asociate, iar *block time*-ul va fi foarte lung. Acest procest mai este cunoscut și sub numele de *mining*.

Proof of Stake sau PoS este cea mai comună alternativă pentru PoW. Sistemele de tip PoS au fost dezboltate să rezolve o parte dintre ineficiențele și problemele care reies din mecanismul PoW. În contextul sistemelor de tip PoS nu se mai discută de *mining*

în sensul vechi, securizarea versiunii fiind deterministă prin faptul că nodul care înscrie noul *block* este ales în baza cantităților de crypto-monedă pe care le pun la dispoziție participanții, unde cei ce dispun mai mult au șanse mai mari să fie aleși. Procesul prin care nodul din rețea ales înscrie noul *block* este cunoscut ca *forging*.

Este considerat de asemenea că acest mecanism este potențial mai rezistent la *51% attacks* deoarece un agent rău intenționat trebuie să aibă în proprietatea lui cel puțin 51% din cantitatea totală de crypto-monedă de pe piață și eșecul atacului implică pierderi financiare imense.

Delegated Proof of Stake sau DPoS este un algoritm de conses dezvoltat de Daniel Larimer în 2014 prin care participanții votează pentru alegerea delegaților care securizează rețeaua în locul lor. Delegații, cunoscuți și sub denumirea de martori, sunt responsabili pentru atingerea consensului în timpul generării și validării noilor *block*-uri. Puterea votului este proporțională cu cantitatea de crypto-monedă deținută.

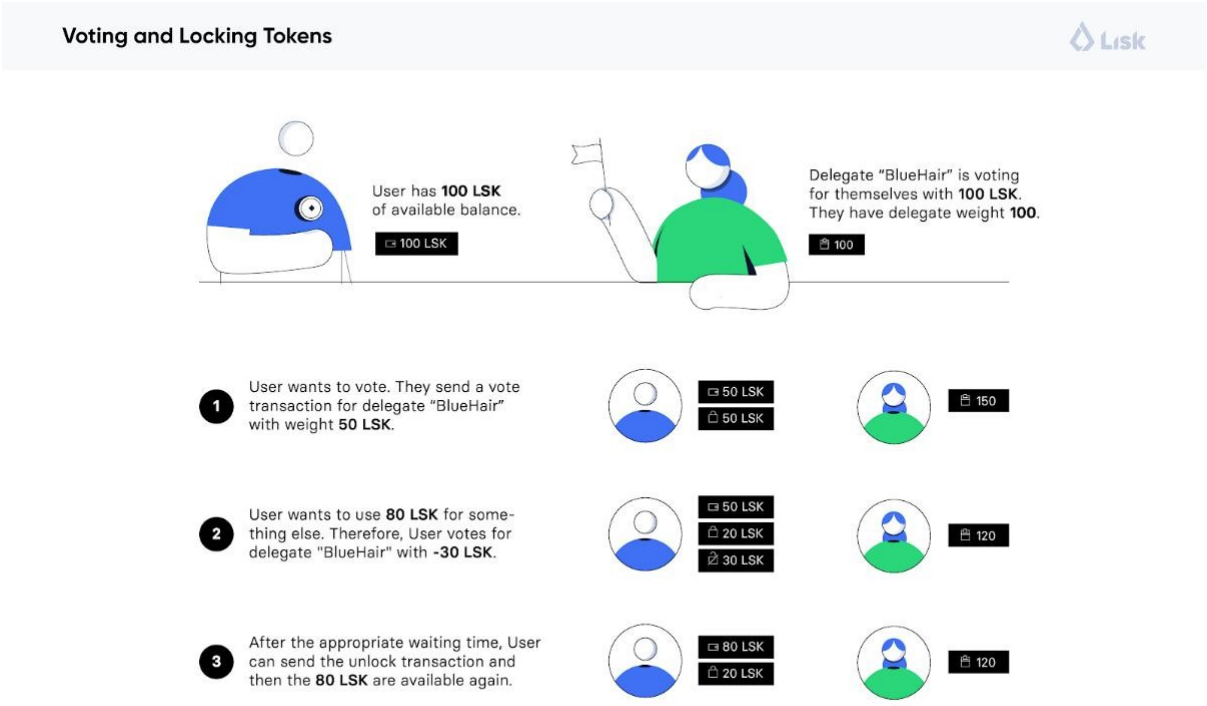

Figura 5: Elecția Delegaților (Lisk, 2019)

În prezent numărul de delegați este fixat la 101, dar acest număr va fi mărit în viitor.

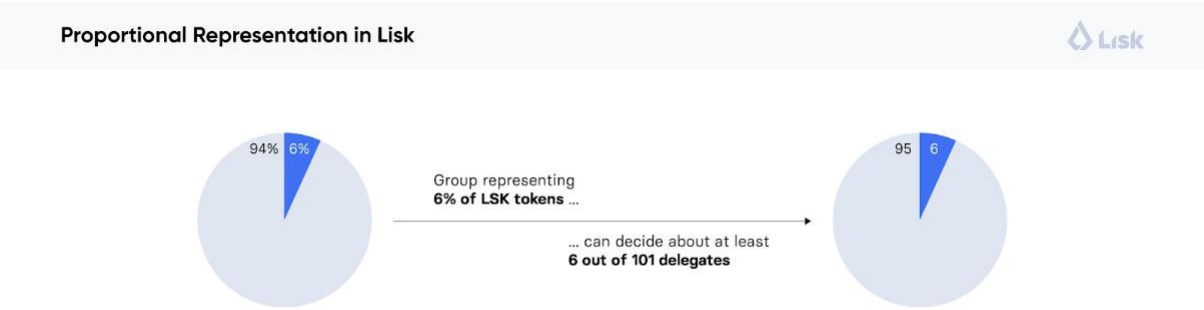

Figura 6: Propoția Elecției (Lisk, 2019)

IPFS

Interplanetary File System sau IPFS este un protocol de rețea peer-to-peer pentru stocarea și diseminarea datelor de orice natură într-un sistem de fișiere distribuit folosind principiul de *content-addressing* pentru a identifica unic fiecare fișier într-un *namespace* global care conectează toate dispozitivele participante.

IPFS este construit în jurul unui sistem descentralizat de utilizatori operatori care dețin o porțiune din toate datele stocate în rețea, astfel creeând un sistem rezilient de stocare și disminare de fișiere. Orice utilizator din rețea poate să servească un fișier și oricare alt utilizator poate să acceseze fișierul respectiv în rețea prin adresa de conținut care este regăsită folosind un *Distributed Hash Table* sau DHT.

IPFS devine unul dintre subsistemele majore ale internetului și dacă este construit și implementat cum trebuie în timp poate să complementeze sau chiar să înlocuiască HTTP. Obiectivul principal al proiectului este să elimine dependența față de *backbone*-ul internetului pentru a creea o infrastructură complet descentralizată și rezilientă care să nu piardă informație pentru că unul dintre sistemele centrale care ține o bucată de conținut pică.

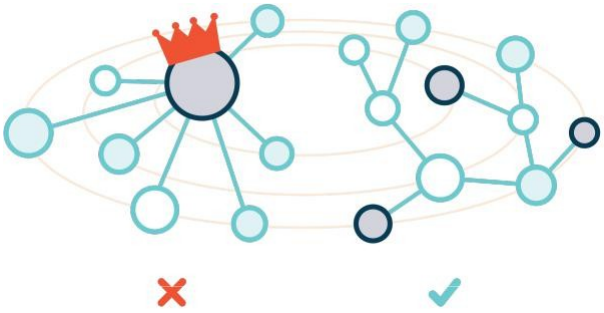

Figura 7: Centralizare v. Descentralizare (IPFS, 2020)

Perspective

Acest sistem poate fi privit în mai multe chei:

IPFS ca protocol:

- Definește un sistem de fișiere adresabil prin hash-ul conținutului efectiv
- Coordonează livrarea conținutului
- Combină conceptual Kademlia + BitTorrent + Git

IPFS ca sistem de fișiere:

- Are directoare și fișiere
- Este montabil la alte sisteme de fișiere via FUSE (*FileSystem in Userspace*)

IPFS ca sistem web:

- Poate fi folosit pentru a consuma documente ca web-ul convențional
- Fișierele sunt accesibile via HTTP prin gateway-uri publice sau private care fac conversia între protocolul standard HTTP și IPFS
- Poate fi folosit pentru a hosta și disemina aplicații statice
- Browserele pot fi extinse pentru a suporta protocolul via ipfs-companion
- Garantează autenticitatea conținutului prin addresarea bazată pe hash-uri

IPFS ca CDN (Content Delivery Network):

- Un fișier adăugat în nodul local este accesibil la nivel mondial
- Implementează un sistem de distribuție de lățime de bandă ca BitTorrent
- Aplică caching în puncte critice sau local pentru a permite conținutului să fie accesibil mai rapid la a doua descărcare

Crypto-adresare

IPFS se folosește de conceptul de *Merkle Directed Acyclic Graphs* sau *MerkleDAG* ca să adreseze criptografic fișierele. Un MerkleDAG este un sistem criptografic prin care *hash*-urile ‘părinte’ își modifică signatura în baza *hash*-urilor ‘copil’ rezultatul fiind o structură arborescentă care poate să fie parcursă într-o manieră scalabilă.

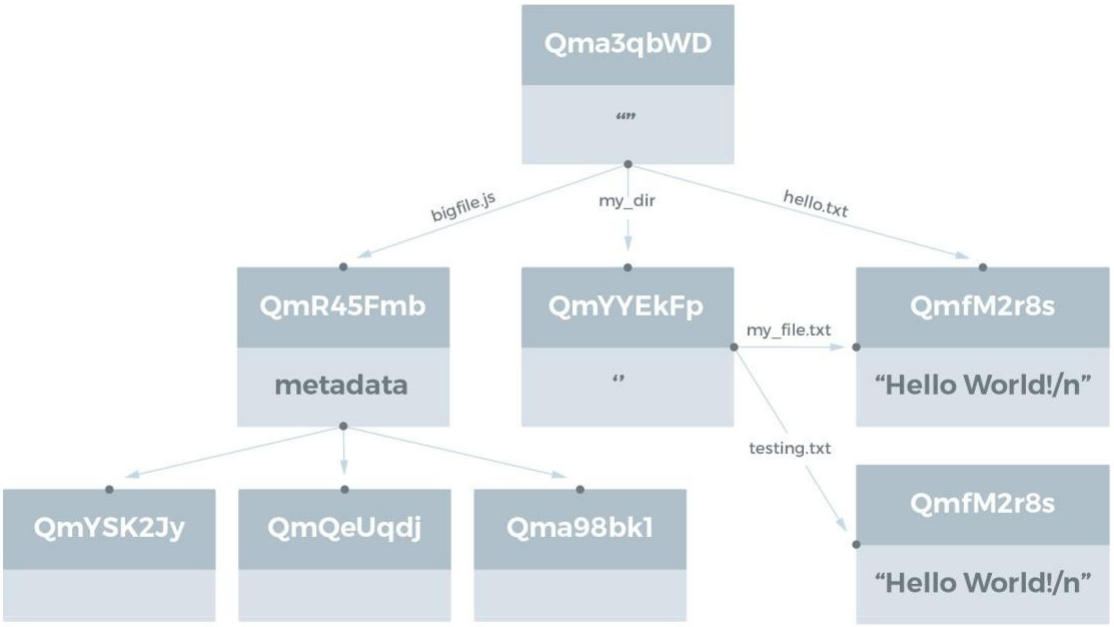

Figura 8: DAG Filesystem (ConsenSys, 2016)

Din exemplul de mai sus conținutul fișierului *my_file.txt* poate să fie adresat în două moduri:

- Fie direct prin *Content Identifier*-ul sau CID-ul lui propriu

```
#!/bin/sh
ipfs cat QmfM2r8seH2GiRaC4esTjeraXEachRt8ZsSeGaWTPLyMoG
#Result: Hello World!
```

- Ori prin CID-ul unui director arbitrar cu *path*-ul relativ concatenat la final

```
#!/bin/sh
ipfs cat Qma3qbWDGJc6he3syLUTaRkJD3vAq1k5569tNMbUtjAZjf/my_dir/my_file.txt
#Result: Hello World!
```

Cum a fost descris în introducere, aplicația prezentată permite utilizatorului să creeze cripto-active care pot fi valorificate speculativ în interiorul rețelei.

Poate să introducă *passphrase*-ul lui personal pentru a intra în aplicațîe cu contul lor asociat sau pot să genereze un passphrase nou dacă nu au mai folosit aplicația până acum.

După logare utilizatorul poate să intre pe pagina portofoliului lor personal și de acolo au posibilitatea să creeze, modifice, distrugă crypto-active sau să creeze oferte de vânzare cu ele.

Pe pagina de explorare utilizatorul poate să vadă toate ofertele de vânzare de cripto-activ valabile pe platformă.

Pachetul final cuprinde toate componentele necesare funcționării unui nod din rețeaua descrisă. Componentele fiind:

- Aplicația frontend Angular 2+ cu care interacționează utilizatorul
- Server-ul NestJS în care se definește API-ul sistemului
- Server-ul Lisk care gestionează componenta blockchain de care se folosește API-ul sistemului împreună cu baza de date PostgreSQL pe care se menține persistența
- Nodul IPFS prin care se realizează expunerea prin rețeaua globală IPFS a informației cu greutate care nu poate fi ținută eficient direct pe blockchain

Aplicația frontend este una standard. Cum a fost descris mai devreme în introducere, diferența între aplicațiile din noua paradigmă nu trebuie să se regăsească la nivel de funcționalitate așteptată de utilizator. UI/UX-ul trebuie prezintă aceeași importanță ca în oricare altă aplicație. Angular Material este framework-ul de UI/UX folosit, acesta fiind o implementare nativă a principiului de Material Design specificat inițial de Google.
Frontend-ul este de tip SPA cu un sistem de rutare în client, aplicația este încărcată odată, iar după cererile către server sunt limitate la cele de nivel API prin care se obține conținutul dinamic care va fi să fie consumat și tranzacționat.

Server-ul NestJS care definește API-ul sistemului este cel care face podul între celelalte 3 componente ale sistemului. În el este definită lista de endpoint-uri HTTP prin care se face accesul la resursele comune din blockchain-ul Lisk și la resursele expuse prin IPFS.

Server-ul Lisk este conținut și gestionat complet de NestJS, acesta reprezentând decât o serie de servicii care definesc logica tranzacțională care este înregistrată comun în blockchain la momentul bootării sistemului.

Tranzacțiile în nomenclatura sistemului Lisk reprezintă instrucțiunile care pot efectiv să facă adăugiri native la blockchain, acestea sunt cele care definesc proprietățiile operațiunilor (taxa și tipul operațiunii) precum și operațiunile în sine care se vor realiza în jurul adreselor agenților participanți.
Dat fiind faptul că este dorința creatorului ca acest sistem să fie complet open-source având toate nodurile desfășurate folosind un sistem de imagini software containerizate în pofida precompilării în binare care ar putea fi distribuite cu mai multă încredere că acestea nu o să fie ușor modificate, apare întrebarea asigurării securității tranzacționale.
Orice individ care va vrea să folosească platforma va fi perfect capabil să schimbe aproape toate componentele sistemului după placul lui, în afară de tranzacțiile blockchain predefinite.

Tranzacțiile predefinite sunt înscrise în blockchain prin tipul și signatura lor în momentul primei bootări a sistemului. În momentul în care un participant va vrea să se conecteze la rețeaua principală ca un alt nod acesta va moșteni implicit setul predefinit de tranzacții specificate de la seed-node-ul lui de la care a ales să culeagă starea inițială a blockchain-ului în momentul primei lui bootări a propriului lui nod.
Dacă un agent mal-intenționat dorește să înscrie o tranzacție nouă sau să modifice una existentă aceasta nu va fi recunoscută de ceilalți agenți pentru că ori nu a existat în punctul de geneză al blockchain-ului ori signatura tranzacției modificate nu convine cu cea deja înregistrată.
Rezultatul este că toți participanții o să poată să facă modificări la nivel de funcționare în partea de client, interacțiunea client-serverAPI și procesul de fetch-ing IPFS fără a putea să afecteze logica tranzacțională efectivă de comun acord a întregului sistem.

În starea curentă a soluției software este descris un singur tip de tranzacție prin care se realizează toate operațiunile care necesită înscriei în blockchain. Acest lucru este inoptim și reprezintă o potențială breșă de securitate, a fost preferată această cale pentru că a fost cea mai rapidă de implementat. Întaine ca sistemul să fie expus publicului acest lucru trebuie reparat.
Filosofia care soluționează acest lucru este cea care garantează granularitate maximă în fiecare tranzacție în parte. Momentan se folosește o singură tranzacție numită AccountTransaction prin care se face transferul de activ în schimbul unei sume convenite, iar logica escrow este gestionată în afara sistemului tranzacțional nativ Lisk care este o parte de logică care trebuie predefinită într-o tranzacție separată pentru a anula riscul de fraudă.
Este de observat aici că nu s-a apelat la o componentă de borkeraj de ordine de vânzare-cumpărare pentru a elimina orice dependență centrală, fapt ce rămâne în continuare un obiectiv principal.

Cea mai simplă componentă din perspectiva sistemului este cea a nodului IPFS atașat. API-ul descris în serverul de NestJS nu face decât să expună mai departe API-ul IPFS pentru a permite accesul la resursele bogate care reprezintă obiectul tranzacției în baza adreselor de localizare după conținut înregistrate în fiecare activ din portofoliul agentului.

Activul este definit momentan în sistem după structura:

```
export class Asset {

	name: string;
	description: string;
	cid: string;
	type: AssetType;
	issued: number;
	value: number;
	tags: string[];
	assetHash: string;
	transactionTimestamp: string;
	content?: Buffer;

	constructor(assetObject: Partial<Asset>={}) {
		this.name =assetObject.name || '';
		this.description =assetObject.description || '';
		this.cid =assetObject.cid || '';
		this.type =assetObject.type || AssetType.NotStonks;
		this.issued =assetObject.issued || 0;
		this.value =assetObject.value || 0;
		this.tags =assetObject.tags;
		this.transactionTimestamp =assetObject.transactionTimestamp || '';
		this.hashAsset()
		.then((data) =>this.assetHash =data)
		.catch((err) =>console.log(err));
	}

	async hashAsset(): Promise<string>{
		let buffer: string ='';
		let hashedProps =[
			this.name,
			this.description,
			this.cid,
			this.transactionTimestamp
		];
		// buffer.concat(hashedProps.values);
		let hash =await sha256(JSON.stringify(this));
		return hash;
	}
}
```

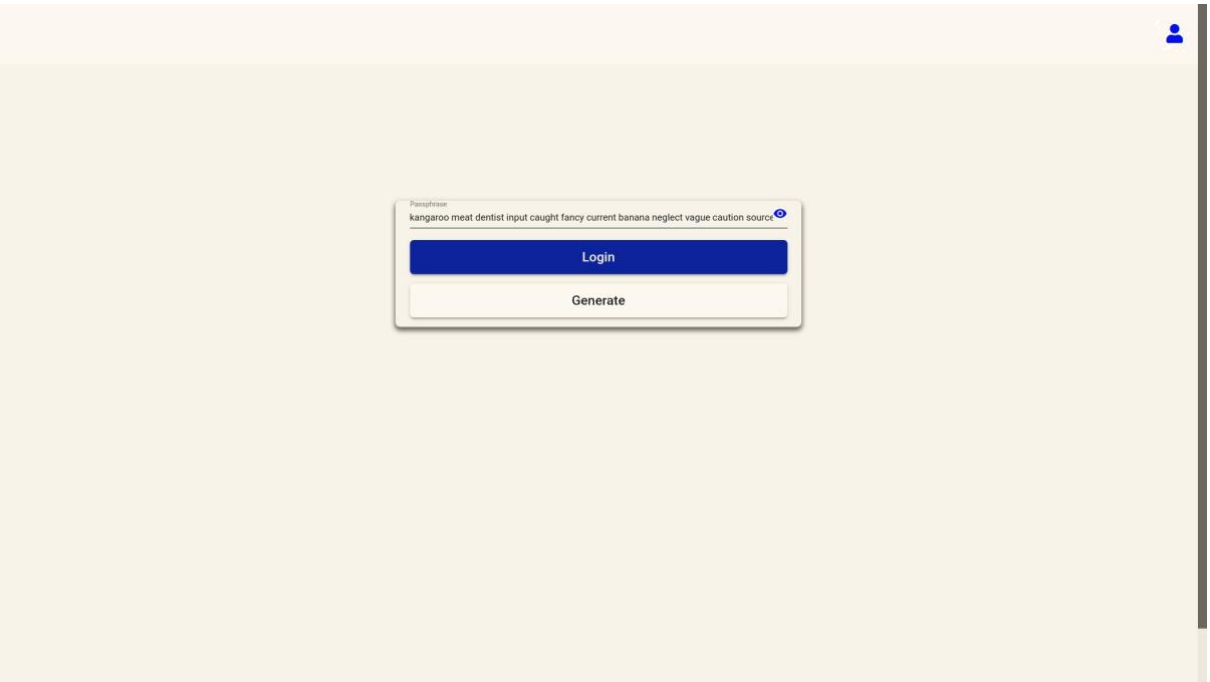

Figura 9: Login: Generare Passphrase

Se observă că pe pagina de Login utilizatorul nu are decât un singur câmp de completat, cel de passphrase cu care să se autentifice. Dacă utilizatorul este nou venit pe platformă tot ce trebuie să facă este să genereze un passphrase nou pe care să-l rețină pe veci.

Informația existenței utilizatorului nu apare în sistem decât în momentul când se face prima operațiune tranzacțională în sistem care se folosește în orice formă de adresa publică derivată din passphrase-ul creat pe pagina de Login.

Butonul “Generate” nu face decât o cerere către CryptoService din serverul API NestJS pentru generarea unui passphrase compatibil cu modulul *@liskhq/lisk-cryptography*.

În momentul în care este acționat butonul de Login nu se face decât o încărcare a passphrase-ului în memoria persistentă a aplicației client pentru reutilizare de abia în momentul când este necesară o operațiune care trebuie înscrisă în blockchain și se dă permisiune utilizatorui să accese următoarele pagini.

Când se face prima cerere pentru înscriere în blockchain, de exemplu când se creează un crypto-activ în portofoliul agentului, atunci se face un request către serverul API NestJS cu passphrase-ul din memoria clientului împreună cu datele relevante și este spart în perechea de chei unice asociate passphrase-ului respectiv pentru a se identifica intrarea nouă în blockchain cu cheia publică a agentului și a se autoriza tranzacția prin semnare folosind cheia privată.

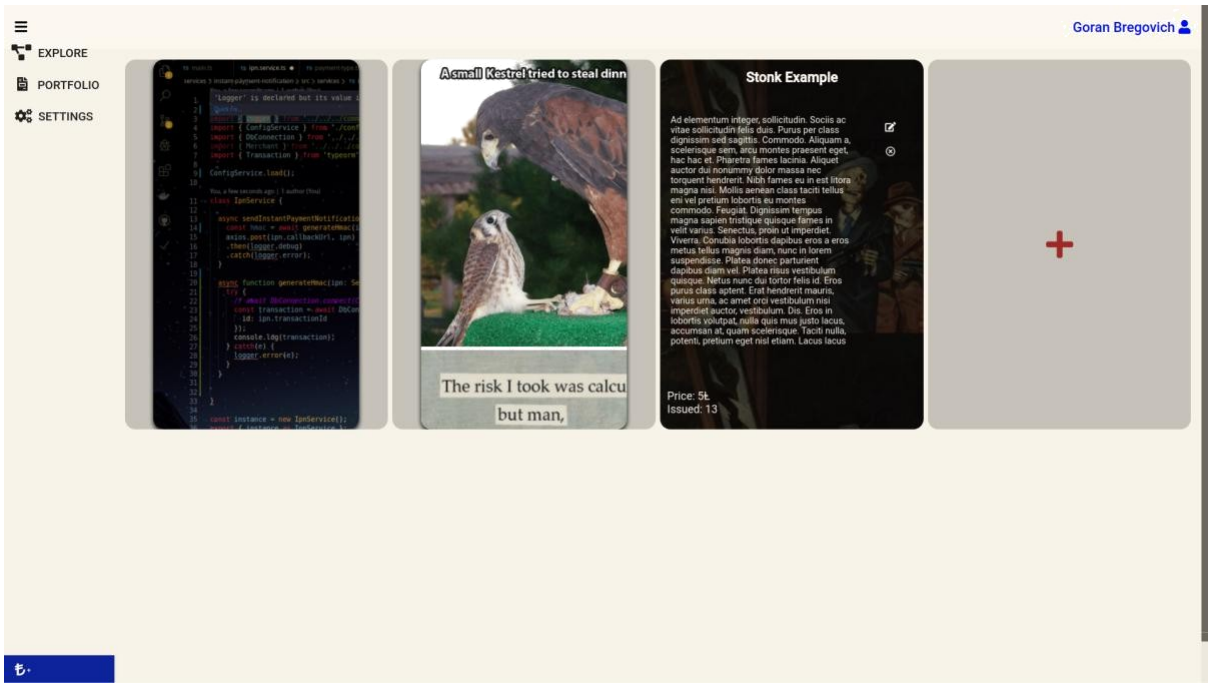

Figura 10: Informații Cripto-Activ

Pe pagina portofoliului utilizatorului se regăsesc cripto-activele care aparțin agentului respectiv fie create de el fie achiziționate de la alți agenți participanți și pe aceeași pagină se realizează procesul de creeare a unui nou ciypto-activ asociat portofoliului în cauză folosind butonul de pe ultima poziție din listă care acționat deschide un modal în care acesta utilizatorul poate să înregistreze datele asociate.

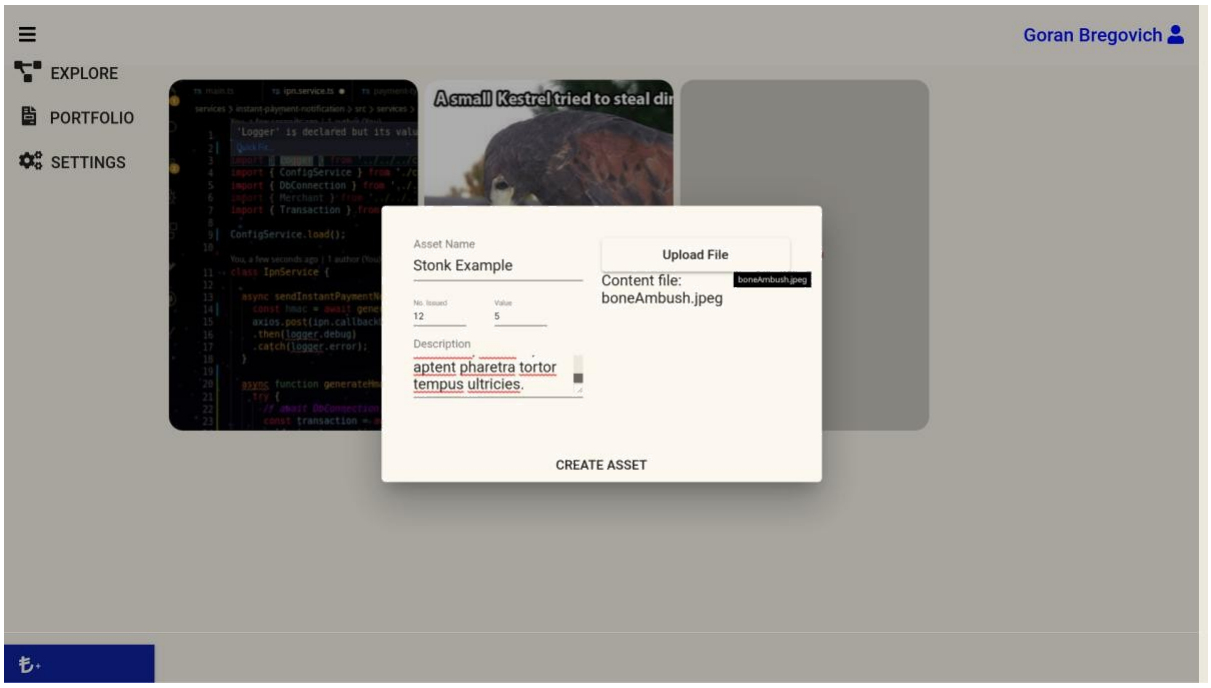

Figura 11: Modal creeare Cripto-Activ

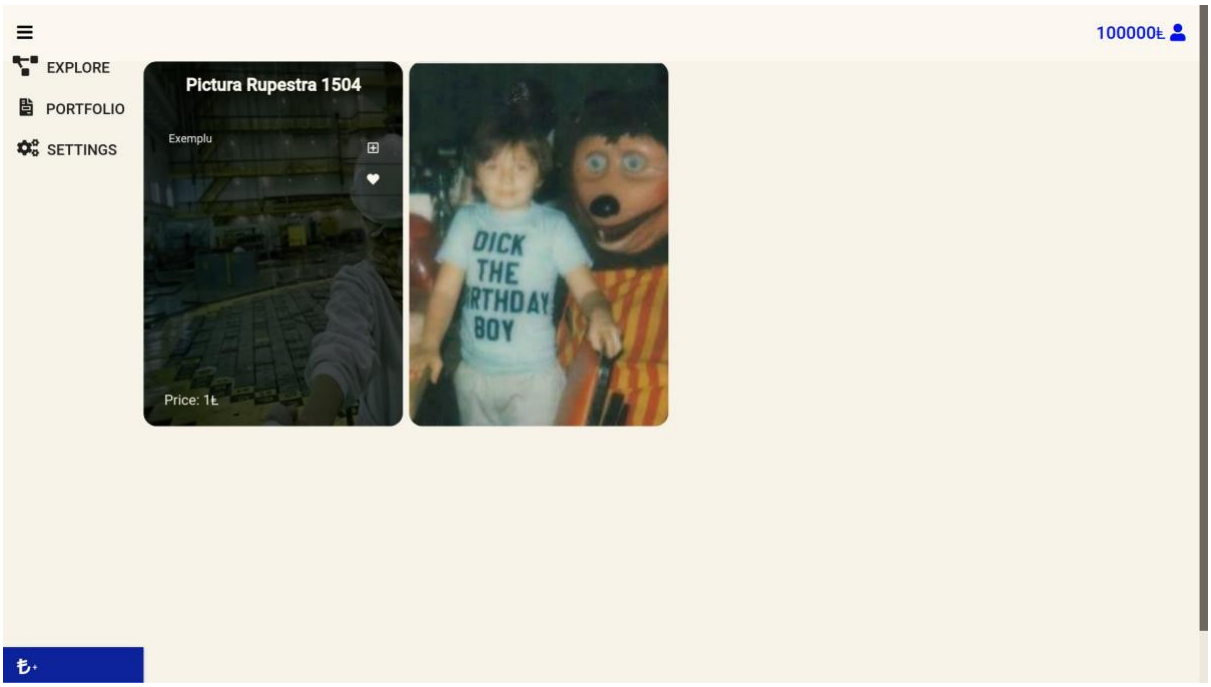

Figura 12: Pagina de explorare

Pagina de explorare este locul în care utilizatorul are posibilitatea să vadă toate ofertele de vânzare de cripto-active existente în piață.

Intrările care apar aici sunt citite direct de pe disk-ul hostului pe care funcționează nodul pe care-l folosește utilizatorul ca să participe la piață. Noile intrări expuse de al ceilalți utilizatori apar în momentul în care este adăugat un nou block de tranzacții în blockchain. Acest lucru se întâmplă odată la 10 secunde care este block time-ul mediu al sistemului Lisk.

Specificaţiile API
Specificațiile endpoint-urilor folosite în serverul API NestJS.

LiskController:

METODA HTTP	ENDPOINT	CERERE	RĂSPUNS
GET	/api/lisk/generate-keys	-	Passphrase
POST	/api/lisk/enrichPass	Passphrase	Passphrase + Perechea de Chei
POST	/api/lisk/account	AccountObject	AccountObject
POST	/api/lisk/updateUser	AccountObject	AccountObject
POST	/api/lisk/sell-order	SellOrder	Rezultat
POST	/api/lisk/buy-order	BuyOrder	Rezultat
GET	/api/lisk/orders	-	List<SellOrder>

IpfsController:

METODA HTTP	ENDPOINT	CERERE	RĂSPUNS
GET	/api/ipfs/asset/:cid	cid	Fişier
GET	/api/ipfs/destroyAsset/:cid	cid	Rezultat
POST	/api/ipfs/store	Binar	Rezultat

API-ul descris mai sus este momentan definit strict pentru buna funcționare a sistemului în starea curentă. El este menit în schimb să fie extins în timp pentru a permite oricărui agent să se integreze programatic cu piața.

Sunt planificate varii endpoint-uri noi de monitorizare granulară care să permită oricărui inginer să analizeze statistic piața pentru a creea proprii lor algorimti automați de vânzare-cumpărare după modelul unei piețe clasice de capital care oferă genul acesta de libertate participanților.
În prezent facilitarea integrării programatice este la fel de importantă ca uzabilitatea manuală prin client-ul de frontend pentru orice platformă de tranzacționare de active.

Procesul de finisare în acest sens ar fi unul în care este necesar feedback-ul inginerilor care încearcă să folosească API-ul pentru scopurile lor proprii și adaptarea la nevoile lor.

Fluxul de valoare și votențialul de vtilizare al soluției

Figura 13: Lisk Protocol Roadmap (Lisk, 2019)

Momentan Lisk de abia a ieșit din stadiul de Alpha și nu are toate funcționalitățile necesare pentru a completa întreg procesul necesar pentru introducerea de valoare în interiorul pieței descrise în lucrarea prezetă, în schimb acestea sunt planificate, iar având în vedere progresul constant pe care l-a manifestat Lisk ca organizațîe în decursul a patru ani este legitim presupus că nu o să lase proiectul să moară.

Fluxul general de valoare pe care îl va avea *dApp*-ul va fi acesta:

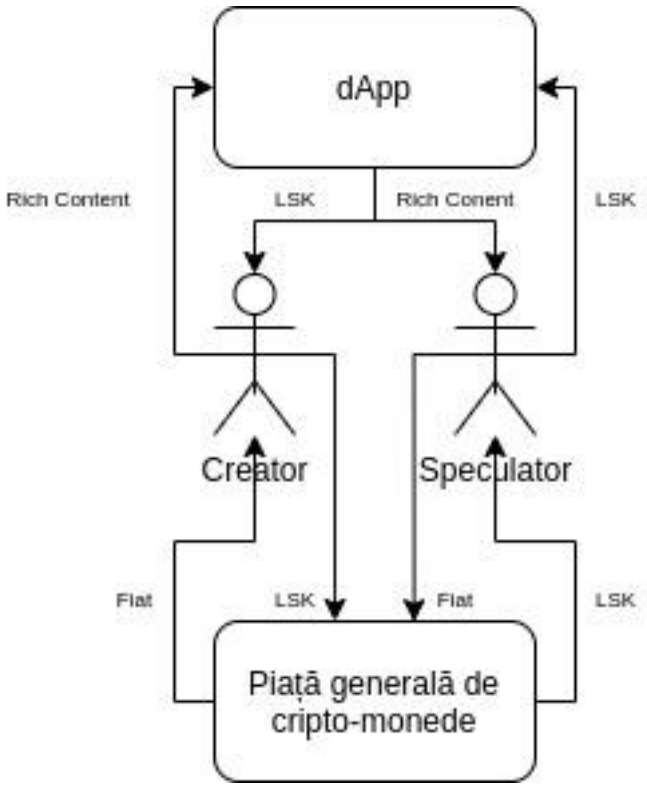

Figura 14: Fluxul de Valoare

Scopul final al platformei va fi mai larg decât cel descris în această lucrare, aici fiind descris mai degrabă doar mecanismul principal de valorificare care are potențialul să reprezinte ca subsistem partea determinantă dintr-un algoritm de rating pentru postări de natură nedeterminată într-o aplicație generală de content sharing și transfer de proprietate cu funcționalități tipice de social media.

În continuare merită reiterat faptul că nu trebuie să existe diferențe la nivel de aplicație între paradigma nouă și cea veche, problematica fiind discutată la nivel arhitectural și infrastructural. Trendurile de design la nivel de aplicație trebuie în continuare respectate, iar așteptările utilizatorilor trebuie în continuare să fie satisfăcute cu prioritate. Având în vedere în schimb că practicile în dezvoltarea aplicațiilor comune de tip social media sunt deja cristalizate acest lucru poate deja să fie considerat trivial.

Funcționalități precum:

- Postări
- Comentarii
- Like-uri (cuantificatori subiectivi)
- Favorite
- Timeline-uri
- Follow
- Pagini Speciale

Pot fi ușor integrate cu sistemul descris fără diferențe aparente.

O platformă similară cu cea planificată în final este LBRY.
LBRY se descrie ca fiind prima piață digitală controlată în totalitate de participanți prin protocolul cu același nume care permite creearea aplicațiilor care se integrează cu restul pieței.
Aceasta deja dă semne de viabilitate, autosuficiență, reziliență și rezistență la cenzură fără a compromite funcționalitățile cu care sunt obișnuiți utilizatorii. Dintre toate încercările de până acum LBRY arată ca cel mai promițător efort de până acum. [14] [15]

Diferențele între proiectul propus aici și LBRY sunt însă în de ajuns de diverse încât să merite continuarea explorării conceptului.

LBRY vine cu propriul său sistem blockchain și propriul protocol de integrare, acest lucru însemnând că un inginer care dorește să ia parte din ecosistem trebuie să învețe sistemul creat de LBRY, fapt care indiferent de eleganța protocolului necesită efort suplimentar. Proiectul propus încearcă să folosească cât mai multe soluții existente, Lisk a fost ales ca sistem de blockchain exact pentru motivul că nu a necesitat un efort de învățare a unui nou ecosistem care este bun numai în contextul lui propriu. Lisk având baza în ecosistemul JavaScript și folosind PostgreSQL ca soluție de persistență a permis o dezvoltare rapidă și scalabilă cu aplicabilitate extinsă și este un sistem de blockchain general în loc de unul concentrat pe un scop ultra specific unui anumit tip de platformă.

Lucrul acesta este evidențiat de poziția celor două cripto-monede în piețele de schimb, la momentul scrierii lucrării LBRY Credits (LBC) sunt listate pe locul 219 [16], iar Lisk (LSK) este listat pe locul 55. [17]

Proiectul propus vine pe lângă ce există deja cu ideea de transfer de propietate asupra conținutului, în sensul că ce într-o aplicație social media clasică, ce ar fi considerată o simplă postare care poate produce bani numai prin donări asociate în forma unui tip special de “like” poate fi în loc privită ca un activ transferabil cu valoare fluctuantă care poate să fie deținut de unul, altul sau mai mulți participanți care fac speculă pe valoarea postării-activ în timp.

Dacă sistemul de rating este bine integrat cu acest proces natural de valorificare poate să mărească orizontul de posibilități pentru felul în care este ordonată relevanța postărilor. În filosofia clasică este presupus că informația cu relevanță maximă pentru utilizator este o funcție dependentă cel puțin doi factori importanți, noutatea și voturile de valoare subiectivă în forma like-urilor sau a upvotes-urilor îmbrăcate în varii algoritmi, dar cu timpul fiind luat ca factor primar.

Această filosofie nu tratează în schimb nevoia ocazională a consumatorului să regăsească ce odată a fost relevant, dar acum e obfuscat de trecerea timpului. Multe postări sunt relevante numai la timpul lor pentru că tratează evenimente care au sens numai în contextul lor temporal, dar de exemplu o poezie care tratează subiectiva condiției umane este decuplată de timp, poate să merite să fie reanalizată în timp din varii motive nedeterminabile, o idee împachetată astfel transcede contextul temporal.

Este viziunea creatorului că un hibrid între algoritmii determinării relevanței folosiți în prezent și un sistem de valorificare memetică bazat pe speculă într-o piață liberă ar atenua fenomenul de irelevanță prematură prin simplul fapt că participanții sunt invitați să investească valoare reală în activul-concept efectiv și chiar să asume proprietate pe varii termeni de timp.

Razultatul este un sistem prin care omul are posibilitatea să atribuie valoare obiectivă fiecărei bucăţi de informaţie sub orice formă, iar conceptele devin separate de autori, ideile plutesc independent în substanţa realităţii.

Un sistem de rating care se foloseşte de factorii clasici, dar primează cu cel al valorii obiective cuantificată în monedă efectivă are potenţialul de a invita consumatorul să se gândească mai adânc la ce este cu adevărat de valoare pentru el şi ca acele lucruri să devină rezistente la trecerea timpului.

Monetizare

Există mai multe opţiuni de strategii de monetizare aplicabile acestui tip de platformă, dintre care ne putem gândi la următoarele care ies în evidenţă imediată:

- Taxă pe fiecare tranzacţie
- Taxă pe anumite tranzacţii specifice
- Advertising în aplicaţia client
- Donări şi sprijin colectiv în formă de crowd-funding

Deşii toate aceste opţiuni sunt viabile în termeni stricţi de business, nu este clar însă impactul pe care fiecare dintre aceste opţiuni îl va avea asupra demograficii ţintă în decursul timpului. O taxă pe fiecare sau câte o anumită tranzacţie nu este momentan adoptată universal în sfera aceasta. LBRY de exemplu în mod explicit a evitat să impună taxe pe tranzacţii şi în continuare caută metode de monetizare în afara ICO-ului iniţial şi specula pe care o fac în continuare pe propria lor monedă, în prezent reuşind totuşi să nu intre în pierdere. [18]

Advertising-ul în interiorul aplicaţiei client deşii ar reprezenta o sursă de venit foarte mare, probabil cu cel mai mare volum potenţial, poate fi privită în fază iniţială ca o trădare a principiilor menţionate în introducerea acestei lucrări de către entuziaştii care o să folosească iniţial acest tip de platformă, independenţa de entităţi terţe fiind un obiectiv principal al motorului ideologic descris în introducere. Acest lucru poate să se schimbe în timp când baza de utilizatori se măreşte şi ajunge să includă indivizi care nu au aceleaşi principii care să le fie încălcate şi să fie promovat un compromis prin care aplicaţia client standard să aibă advertising, iar cei ce nu vor dori acest lucru o să aibă posibilitatea de a folosi un client alternativ creeat de altă entitate care s-a integrat programatic cu piaţa via API-ul expus de nod.

În final cea mai curată soluţie o să fie cea a sprijinului colectiv sub forma de crowd-funding, această strategie devenind din ce în ce mai viabilă cu fiecare an, utilizatorii devenind obişnuiţi cu ideea de sprijin voluntar direcţionat către ce valorează pentru el cel mai mult.

Concluzii

Platforma a fost inițial planificată ca o piață liberă de conținut a cărui valoare este securizată de mecanismele intrinseci unui sistem blockchain. Potențialul platformei este mai larg de atât.

Platforma în starea curentă este capabilă de distribuire informațională complet descentralizată. Toate componentele necesare funcționării apicației ca nod în rețea sunt cuprinse într-un singur pachet distribuibil într-o formă ușor de instalat pe orice dispozitiv uzual, iar acesta va fi singurul lucru pe care un prospectiv participant îl va trebui face pentru a deveni un nod în rețea și a participa pe platformă. Toate nodurile din rețea au aceleași drepturi la nivel de logică relevantă platformei, nu există noduri speciale care să aibă mai mult control asupra proceselor specificate, nici măcar creatorul platformei nu va avea mai multe drepturi decât utilizatorul normal.

Într-o lume în care oligopolul companiilor care dețin puținele aplicații sociale, care au ajuns să reprezinte surse primare de informație pentru mare parte din public, tinde să se manifeste prin cenzură selectivă și colectare clandistină de date fie prin propria lor voință sau împinse de forțele coercitive ale statelor lumii, foamea de transparență, anonimitate și independență crește în zeitgeistul contemporan pe zi ce trece.

Aplicația prezentată poate să fie ușor extinsă pentru a cuprinde toate funcționalitățiile tuturor *site*-urilor și aplicațiilor pe care le folosește publicul în mod uzual, iar faptul că acest lucru se poate realiza fără niciun cost de mentenanță pentru complexul de servere mari care trebuie să conțină toată informația distribuită pe platformă face ca acest lucru să fie mai mult decât facil. Costul este distribuit uniform între utilizatori și nu există nicio entitate centrală care să fie necesară pentru a ține platforma în picioare, rețeaua întreagă având potențial de autosuficiență deplină.

În afara unei serii de evenimente dramatice care să schimbe masiv cursul istoriei, este foarte probabil că aceasta va fi direcția generală a dezvoltărilor tehnologice în cea de a treia eră Web.

Lista figurilor utilizate

Bibliografie

[1] - (2020), “AJAX”„ https://en.wikipedia.org/wiki/Ajax_(programming).

[2] - (2018), “Article 13 of the copyright directive raises serios questions”„ https://www.internetsociety.org/blog/2018/06/article13/?gclid=CjwKCAjwvtX0BRAFEiwAGWJyZFwtiG5Q1uiITy9qpprzWMrwbRK597_4xzOYbD0amLOs5-RSvRvYnRoC1rAQAvD_BwE.

[3] - (2020), “A sneaky attempt to end encryption is worming it’s way through congress”„ https://www.theverge.com/interface/2020/3/12/21174815/earn-it-act-encryption-killer-lindsay-graham-match-group.

[4] - (2020) , “Common gateway interface”„ https://en.wikipedia.org/wiki/Common_Gateway_Interface.

[5] - ConsenSys. (2016), “An introduction to ipfs”, https://medium.com/@ConsenSys/an-introduction-to-ipfs-9bba4860abd0.

[6] - Diagoal. (2018), “Lisk”, https://yakucoding.com/lisk/.

[7] - Downey, L. (Ed.). (2020), “Store of value - definition”, https://www.investopedia.com/terms/s/storeofvalue.asp; Investopedia.

[8] - Frankenfield, J. (2019), “Block time”, https://www.investopedia.com/terms/b/block-time-cryptocurrency.asp.

[9] - Hackfeld, J. (2019), “3 new dpos lips: Changing the voting system for lisk”, https://lisk.io/blog/research/3-new-dpos-lips-changing-voting-system-lisk.

[10] - Lisk. (2020), “Protocol roadmap”, https://lisk.io/roadmap.

[11] - Wikipedia (2020)“Server side includes”„ https://en.wikipedia.org/wiki/Server_Side_Includes.

[12] - Sharma, M. (2019), “Web 1.0, web 2.0 and web 3.0 with their difference”, https://www.geeksforgeeks.org/web-1-0-web-2-0-and-web-3-0-with-their-difference/; GeeksforGeeks.

[13] – Jason Bailey (2018), “Rare Pepe Wallet & The Birth Of CryptoArt”,, https://www.artnome.com/news/2018/1/23/rare-pepe-wallet-the-birth-of-cryptoart#:~:text=Rare Pepe Wallet is a,gift%2C and destroy digital artworks.&text=First to create a gift,innovation for improving adoption)%3B

[14] – LBRY (2020), “What is LBRY exactly? Is it a protocol, an app, a website, or a company?”

[15] – LBRY (2020), “Art in the Internet Age”,, https://lbry.com/what

[16] – CoinRanking (2020), “LBRY Credits (LBC) price to USD & live value of today”,,
https://coinranking.com/coin/kVEZYPoF-nPPH+lbrycredits-lbc
[17] – CoinRanking (2020), “Lisk (LSK) price to USD & live value of today”,,
https://coinranking.com/coin/-8WmRrc-b_6dN+lisk-lsk
[18] – LBRY (2020), “How does the company behind LBRY make money?”,,
https://lbry.com/faq/lbry-revenue

Anexă

Configurarea și inițializarea modulului Lisk

```
import { Module, HttpModule } from '@nestjs/common'; import {
LiskService } from './lisk.service'; import { LiskController } from
'./lisk.controller'; import { CryptoService } from './crypto.service';
import { getNetworkIdentifier } from '@liskhq/lisk-cryptography';
import { Application, genesisBlockDevnet, configDevnet } from 'lisk-sdk'; import
{ APIClient } from '@liskhq/lisk-api-client'; import { Logger } from '../../../common/logger';
import { AccountTransaction } from './transactions/account.transaction'; import
{ DEVNET_URL } from '../../../common/env_vars';
import { OrderTransaction } from './transactions/order.transaction';

configDevnet.modules.chain.forging.force = true;

genesisBlockDevnet.transactions[0] = {
   id: '7646387794267587684',
   type: 8,
   timestamp: 0,
   senderPublicKey:
   'edf5786bef965f1836b8009e2c566463d62b6edd94e9cced49c1f098c972b92b',
   signature:
   '9f1282585cf91c9da0355f8e75c53363e50c0c1d41e96756b2bda02991ecb351bf67...',
   asset: { amount: '10000000000000000', recipientId: '16313739661670634666L' }
}

export const lisknet = new APIClient(
   process.env.MAINNET ? APIClient.constants.MAINNET_NODES :
   APIClient.constants.TESTNET_NODES
);
```

```
export const devnet = new APIClient(
   [DEVNET_URL]
);

export const networkIdentifier = getNetworkIdentifier(
   "23ce0366ef0a14a91e5fd4b1591fc880ffbef9d988ff8bebf8f3666b0c09597d", "Lisk",
);

@Module({
   imports: [HttpModule],
   providers: [{
         provide: 'LISK_SETUP',
         useFactory: () => {
            const app = new Application(genesisBlockDevnet, configDevnet);
            app.registerTransaction(AccountTransaction);
            app.registerTransaction(OrderTransaction); app

            .run()
            .then(() => console.log('Lisk Connected'))
            .catch(err => {
               console.error(err);
               process.exit(1);
            });

         }
      },
      LiskService,
      CryptoService
   ],
   controllers: [LiskController]
})
export class LiskModule {}
```

Serviciul principal de logică tranzacțională (Snippets)

```
export function timestamp() {
      const millisSinceEpoc = Date.now() - Date.parse(EPOCH_TIME.toString()); const
      inSeconds = ((millisSinceEpoc) / 1000).toFixed(0); return parseInt (inSeconds)-1;
}

async updateAccount(user: User) {
   const tx = new AccountTransaction({
      timestamp: timestamp(),
      networkIdentifier: networkIdentifier,
      asset: user.asset
   });
   tx.sign(user.passphrase);

   try {
      const result = await devnet.transactions.broadcast(tx.toJSON())
      log.info('Transaction result: ', result);
      return result;
   } catch(err)
      { log.error(err);
      throw new Error("Fail on update account");
   }
}

async transact(buyOrder: BuyOrder, passphrase: string, asset: Asset) { const tx =
   trans.transfer({
      amount: convertLSKToBeddows(buyOrder.offeredPrice.toString()),
      networkIdentifier: networkIdentifier,
      recipientId: buyOrder.sellerId,
      passphrase: passphrase
   });

   try {
```

```
        await devnet.transactions.broadcast(tx);
    } catch(err) {
        throw new InsufficientFunds("That wasn't very cash money of you");
    }

    try {
        await this.addAssetToUserPortfolio(asset, passphrase); } catch(err)
    {
        throw new Error("Failed to transfer asset to portfolio")
    }

    try {
        await this.removeSellOrder(buyOrder.sellOrderId); }
    catch(err) {
        throw new Error("Failed to remove sell order");
    }
    return;
}

async addSellOrder(sellOrder: SellOrder) {
    let orderAccount: OrderAccount = await this.getOrderAccount();
    let asset: Orders = orderAccount.asset;
    asset.sellOrders.push(sellOrder);

    const tx = new OrderTransaction({
        timestamp: timestamp(),
        networkIdentifier: networkIdentifier,
        targetAccountAddress: getAddressFromPassphrase(sellPass),
        asset: asset
    });
    tx.sign(sellPass);

    try {
        const result = await devnet.transactions.broadcast(tx.toJSON());
        log.info('Transaction result: ', result);
```

```
        return result;
    } catch(err)
        { log.error(err);
        return { status: 400, message: "Please try again" };
    }
}

async addBuyOrder(buyOrder: BuyOrder, passphrase: string) {
    let orderAccount: OrderAccount = await this.getOrderAccount();
    let asset: Orders = orderAccount.asset;
    asset.buyOrders.push(buyOrder);

    const relevantSellOrder = asset.sellOrders.filter((sellOrder) => sellOrder.id == bu

    try {
        await this.transact(buyOrder, passphrase, relevantSellOrder.assetData);
    } catch(err)
        { log.error(err);
        throw new HttpException(err.message, 400);
    }

    const tx = new OrderTransaction({
        timestamp: timestamp(),
        networkIdentifier: networkIdentifier,
        targetAccountAddress: getAddressFromPassphrase(sellPass),
        asset: asset
    });
    tx.sign(sellPass);

    try {
        const result = await devnet.transactions.broadcast(tx.toJSON());
        log.info('Transaction result: ', result);
        return result;
        ...
}
```

Exemplu tranzacţie personalizată

```
export class AccountTransaction extends BaseTransaction {

  private accountAsset: any;

  protected verifyAgainstTransactions(transactions: readonly import("@liskhq/lisk-tra throw new
    Error("Method not implemented.");
  }
  protected assetFromSync(raw: any): object { throw
    new Error("Method not implemented.");
  }

  constructor(transObj: TransactionInfo) {
    super(transObj);
  }

  static get TYPE() {
    return 13;
  }

  static get FEE() {
    return `0`;
  };

  async prepare(store) {
    await store.account.cache([
      {
        address: this.senderId
      },
    ]);
  }

  validateAsset() {
    const errors = [];
```

```
        const asset: any = this.asset;

        return errors;
    }

    applyAsset(store) {
        const errors = [];
        const asset: Partial<User> = this.asset;
        let sender: any = store.account.get(this.senderId);

        const newObj = { ...sender, asset: asset };
        store.account.set(sender.address, newObj);

        return errors;
    }

    undoAsset(store) {
        const sender = store.account.get(this.senderId); const
        nullAssetObj = { ...sender, asset: null };
        store.account.set(sender.address, nullAssetObj);
        return [];
    }

    assetToBytes() {
        return Buffer.from(JSON.stringify(this.asset));
    }

    assetToJSON() {
        return this.asset;
    }

}
```