(19) OFICIUL DE STAT
PENTRU INVENŢII ŞI MĂRCI
Bucureşti

ROMÂNIA

(11) RO 134065 A0
(51) Int.Cl.
G06F 21/30 (2013.01);
G06F 21/60 (2013.01);
G06F 21/62 (2013.01);
G06Q 20/38 (2012.01);
H04L 9/06 (2006.01)

(12) # CERERE DE BREVET DE INVENŢIE

(21) Nr. cerere: **a 2019 00496**

(22) Data de depozit: **14/08/2019**

(41) Data publicării cererii:
30/04/2020 BOPI nr. **4/2020**

(71) Solicitant:
• **INGENIUM BLOCKCHAIN TECHNOLOGIES S.R.L.**,
STR.EUFROSINA POPESCU NR.54, CAMERA 1, BL.37A+B, SC.C, ET.5, AP.111, SECTOR 3, BUCUREŞTI, B, RO

(72) Inventatori:
• **IFTEMI ALIN-DANIEL**, *STR.BUCOVINA NR.7, BL.G2, SC.3, ET.3, AP.54, SECTOR 3, BUCUREŞTI, B, RO*

(74) Mandatar:
WEIZMANN ARIANA & PARTNERS AGENŢIE DE PROPRIETATE INTELECTUALĂ S.R.L., *STR.11 IUNIE NR.51, SC.A, ET.1, AP.4, SECTOR 4, BUCUREŞTI*

(54) # PLATFORMĂ ŞI METODĂ DE CONECTARE A UNUI MOTOR DE BLOCKCHAIN

(57) **Rezumat:**

Invenţia se referă la o platformă şi la o metodă de conectare a unui motor de blockchain cu o bază de date tradiţională. Platforma conform invenţiei este implementată sub forma unei reţele de noduri care este împărţită în cel puţin două subreţele: o subreţea de securitate şi o subreţea de date, toate nodurile din subreţeaua de securitate conţinând informaţii referitoare la cheile de securitate ale utilizatorilor, licenţele de funcţionare, politicile de acces, precum şi alte informaţii care ţin de mecanismul de licenţiere, autentificare şi autorizare a utilizatorilor care accesează platforma, iar nodurile din cel puţin o subreţea de date cuprind o componentă software care foloseşte o reţea de calculatoare, o interfaţă de comunicare API ce permite interacţiunea cu reţeaua de calculatoare şi preluarea informaţiilor care trebuie salvate într-un sistem de stocare, o interfaţă GraphQL de interogare a datelor, un motor de procesare a datelor, un motor blockchain, o interfaţă de conectare între motorul de procesare a datelor şi motorul blockchain, şi o bază de date.

Revendicări: 25
Figuri: 7

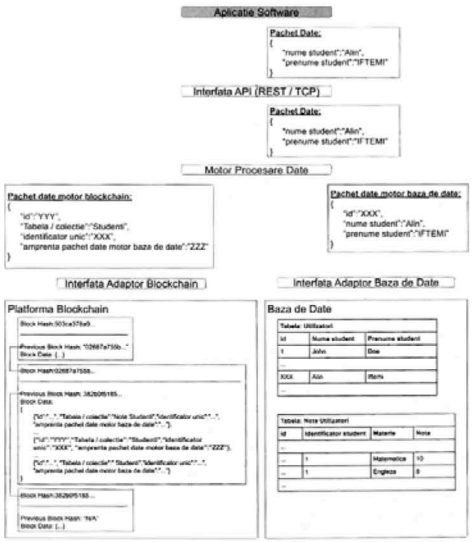

Fig. 3

RO 134065 A0

138

PLATFORMĂ ŞI METODĂ DE CONECTARE A UNUI MOTOR DE BLOCKCHAIN

Descriere

Invenţia se referă la platformă şi la o metodă de conectare destinată să conecteze un motor de blockchain cu cel puţin o bază de date tradiţională.

Un blockchain este o listă de înregistrări (sau date) în continuă creştere, numite blocuri, care sunt legate şi securizate cu ajutorul criptografiei. Ca structură de date, un blockchain este o listă simplu înlănţuită, în care legăturile între elemente se fac prin hash (funcţie hash). Astfel, fiecare bloc conţine de obicei o legătură către un bloc anterior (un hash al blocului anterior), un timestamp şi datele tranzacţiei. Prin design, blockchain-urile sunt rezistente la modificarea datelor. Blockchain-ul este „un registru transparent şi distribuit care poate înregistra tranzacţii între două părţi în mod eficient, verificabil şi permanent". Pentru a fi folosit ca registru distribuit, un blockchain este de obicei administrat de o reţea colectivă de tip peer-to-peer, ce aderă la un protocol pentru validarea noilor blocuri. Odată înregistrate, datele din orice bloc de date nu mai pot fi modificate retroactiv fără alterarea blocurilor care urmează, ceea ce necesită acordul majoritar al participanţilor din reţea.

Blockchain-urile sunt securizate prin design şi sunt un exemplu de sistem de calcul distribuit cu toleranţă ridicată de tip bizantin (toleranţă la atacatori sau la calculatoare necooperante). Problema consensului descentralizat a fost prin urmare rezolvată cu ajutorul tehnologiei blockchain. Acest lucru face ca tehnologia blockchain să fie adecvată pentru înregistrarea de evenimente, dosare medicale precum şi înregistrarea altor activităţi de management cum ar fi gestionarea identităţii, procesarea tranzacţiilor, documentarea provenienţei, urmărirea traseului comercial al produselor alimentare sau sisteme de votare.

Primul blockchain public a fost conceptualizat în anul 2008 de o persoană anonimă care s-a identificat cu numele de Satoshi Nakamoto. În 2009 a fost pus în aplicare în cripto-moneda Bitcoin, unde serveşte ca registru public şi descentralizat pentru toate tranzacţiile. Inventarea tehnologiei blockchain a făcut ca Bitcoin să devină prima monedă digitală care a rezolvat problema dublei cheltuieli fără să se folosească de o autoritate centrală de încredere sau de servere centrale. Designul monedei Bitcoin a fost sursă de inspiraţie pentru multe alte aplicaţii.

Tehnologia blockchain s-a evidenţiat printr-o serie de beneficii tehnice şi conceptuale care se dovedesc a avea un impact foarte mare in ceea ce priveşte guvernarea interacţiunii

parţilor cu produsele software precum si securitate datelor pe care aceştia le operează. Printre acestea enumeram:

- **Imutabilitatea datelor**: datorita structurii de blocuri de date dependente logic prin semnături de tip hash, rezistenta la modificarea datelor este una din principalele caracteristici pe care tehnologia blockchain le aduce la nivel de securitate şi protecţie a datelor stocate
- **Descentralizarea**: eliminarea unei guvernări centrale a datelor şi a utilizatorilor care accesează o aplicaţie bazata pe tehnologia blockchain. Datele sunt stocate intr-un sistem accesibil tuturor parţilor iar utilizatorii îl pot accesa direct în nodul personal, în baza regulilor şi politicilor de securitate fata de care s-au supus. Aceasta proprietate creste nivelul de încredere al utilizatorilor în modul în care sistemul gestionează datele şi protejează confidenţialitatea utilizatorilor.
- **Distributivitatea**: scalabilitatea sistemului este garantata de prezenta mai multor noduri , unde fiecare dintre noduri conţine o copie integrala a datelor (blockchain-ul) precum şi a programului informatic care facilitează accesul la aceste date. Sistemele distribuite nu sunt o inovaţie a tehnologiei blockchain, ele au existat de foarte mult timp, însa tehnologia se diferenţiază prin însumarea mai multor tehnologii existente care împreuna crează o tehnologie inovativa, nu neapărat prin beneficiile aduse de fiecare componenta în parte.
- **Securitatea**: protecţia datelor (criptare/decriptare), dar şi algoritmii criptografici de semnare a tranzacţiilor care sunt înglobate într-un bloc, conferă un nivel înalt de securitate a datelor care nu numai ca garantează protecţia acestora prin criptare dar garantează şi autentificarea, autenticitatea, integritatea şi non-repudierea acestora.

Aplicaţia Bitcoin a stârnit interes comunităţilor de tehnologii prin robusteţea, rezistenta la atacuri informatice şi securitatea prin care aceasta reuşeşte sa funcţioneze fără a fi guvernata de o entitate. Tehnologia blockchain care sta la baza aplicaţiei Bitcoin a devenit extrem de apetisanta, şi s-a dovedit ca beneficiile pe care aceasta le oferă pot fi utilizate şi la nivel de aplicaţii software industriale.

Prezenta invenţie îşi propune să furnizeze o platformă şi o metodă de conectare care să permită implementarea tehnologiei blockchain în soluţiile software pe care le dezvoltă sau pe care le-au dezvoltat deja.

Un alt obiectiv al invenţiei este acela de a facilita adopţia blockchain la nivel industrial, combinând tehnologia blockchain cu mecanismele de stocare de date (motoare baze de date) tradiţionale.

În conformitate cu prezenta invenţie, platforma de conectare este destinată să conecteze un motor de blockchain cu o bază de date tradiţională, platforma fiind implementată sub forma unei reţele de noduri, reţeaua de noduri menţionată fiind împărţită în cel puţin două subreţele: o subreţea de securitate şi o subreţea de date, toate nodurile din sub-reţeaua de securitate conţinând informaţii referitoare la cheile de securitate ale utilizatorilor, licenţele de funcţionare, politice de acces, precum şi alte informaţii care ţin de mecanismul de licenţiere, autentificare şi autorizare al utilizatorilor care accesează platforma, şi în care nodurile din cel puţin o subreţea de date cuprind o componentă software care foloseşte o reţea de calculatoare, o interfaţă de comunicare API ce permite interacţiunea cu reţeaua de calculatoare şi preluarea informaţiilor care trebuie salvate în sistemul de stocare, o interfaţă GraphQL de interogare a datelor, un motor procesare date, un motor blockchain, o interfaţa de conectare între motorul de procesare date şi motorul blockchain, şi o bază de date.

Alte caracteristici preferate ale platformei conform prezentei invenţii sunt prezentate în revendicările dependente 2-11.

Obiectivele invenţiei sunt atinse de asemenea cu ajutorul unei metode metode de conectare a unui motor de blockchain cu cel puţin un motor de management a bazelor de date, metoda cuprinzând etapele de asigurare a unei platforme de conectare, conform revendicării 1, platforma menţionată fiind implementată sub forma unei reţele de noduri împărţită în cel puţin două subreţele: o subreţea de securitate şi o subreţea de date, toate nodurile din sub reţeaua de securitate conţinând informaţii referitoare la cheile de securitate ale utilizatorilor, licenţele de funcţionare, politice de acces, precum şi alte informaţii care ţin de mecanismul de licenţiere, autentificare şi autorizare al utilizatorilor care accesează platforma, şi în care nodurile din acea cel puţin o subreţea de date cuprind o componentă software care foloseşte o reţea de calculatoare, o interfaţă de comunicare API ce permite interacţiunea cu reţeaua de calculatoare şi preluarea informaţiilor care trebuiesc salvate în sistemul de stocare, o interfaţă GraphQL de interogare a datelor, un motor procesare date, un motor blockchain, o interfaţa de conectare între motorul de procesare date şi motorul blockchain, şi o bază de date.

Alte caracteristici preferate ale platformei conform prezentei invenţii sunt prezentate în revendicările dependente 13-24.

Alte obiective, caracteristici şi avantaje ale prezentei invenţii vor reieşi mai clar din următoarea descriere detaliată a unor exemple preferate de realizare a invenţiei, dată în legătură cu Figurile anexate, care prezintă schematic şi nu limitativ conceptele prezentei invenţii, şi în care:

- Figura 1: este o diagramă schematică implementare a nodurilor platformei conform prezentei invenţii;
- Figura 2 este o diagramă schematică a structurii unui nod al platformei conform prezentei invenţii;
- Figura 3 este o diagramă exemplificativă pentru mecanismul de stocare a datelor;
- Figura 4 este o schemă ilustrativă a unei reţelei deschise;
- Figura 5 este o schemă ilustrativă a unei reţele restricţionate (1)
- Figura 6 este o schemă ilustrativă de reţea restricţionată (2)
- Figura 7 este o diagramă exemplificativă de reţea cu un nod FULL NODE şi 2 noduri PARTIAL NODE

Cu referire la Figurile anexate, platforma conform prezentei invenţii este o plarformă de de conectare, o platforma de intermediere, care conectează un motor de blockchain (Ex: Ethereum, Hyperledger) cu o baza de date tradiţionala (Ex: MongoDb, Oracle, MsSql). Aceasta conexiune (hibrid) este îmbunătăţită cu un set de proprietăţi, beneficii şi funcţionalităţi care au ca scop înlesnirea adopţiei de către programatori dar şi uşurarea muncii acestora în ceea ce priveşte dezvoltarea de aplicaţii descentralizate şi cu grad înalt de securitate a datelor.

Platforma conform prezentei invenţii este uşor accesibilă prin intermediul API (Application Programming Interface / Drivers), asa cum majoritatea programatorilor sunt obişnuiţi să interacţioneze cu mecanismele de stocare de date existente.

La nivel arhitectural platforma conform prezentei invenţii (pe parcursul prezentei documentaţii, platforma conform prezentei invenţii poate fi denumită şi HyperFiuse, fiind înţeles faptul că această denumire nu limitează în nici un fel caracteristicile şi domeniul de aplicare al invenţiei) se prezintă sub forma unei reţele de calculatoare (care se mai numesc şi noduri). Aceasta reţea de calculatoare este grupată în subretele, fiind necesare cel puţin două subreţele în crearea unei infrastructuri: sub-reţeaua de securitate şi sub-reţeaua de date. La

rândul ei, sub-reţeaua de date poate fi împărţită în mai multe sub-reţele, în funcţie de zone geografice sau alte grupări logice convenabile administratorului de sistem.

Fiecare sub-reţea conţine unul sau mai multe noduri, tipul de sub-reţea fiind dat de tipul de date stocat pe nodurile care o compun. In sub-reţeaua de securitate, toate nodurile conţin informaţii referitoare la cheile de securitate ale utilizatorilor, licenţele de funcţionare, politice de acces precum şi alte informaţii care ţin de mecanismul de licenţiere, autentificare şi autorizare al utilizatorilor care accesează sistemul.

Mecanismul de autentificare şi autorizare folosit în motoarele de blockchain publice (vezi Bitcoin/Ethereum) se bazează pe cupluri de chei publice/private, dar cheia privata este în posesia utilizatorului. Este bine cunoscut faptul ca pierderea unei chei private, duce la imposibilitatea decriptării informaţiei care a fost criptata cu cheia publica asociata acesteia. Dacă în aplicaţiile de tip blockchain public, regulile sunt foarte clare de la bun început, şi fiecare utilizator îşi asuma păstrarea cheii private, în mediul de business lucrurile stau cu totul altfel. Datorită riscului ridicat pe care îl prezintă pierderea unei chei de decriptare în mediul de business, a fost creata sub-reţeaua de securitate care va stoca aceste informaţii pentru utilizatori. Menţionez ca sistemul poate funcţiona şi prin descentralizarea completa a utilizatorilor si cheilor asociate acestora, în unul din următoarele cazuri:

- Beneficiarul îşi asumă ca utilizatorii să ţină cheile în custodie proprie
- Se conectează un furnizor extern care va face mentenanţa cheilor utilizatorilor
- Se folosesc chei hardware pentru utilizatorii care accesează aplicaţia

Descentralizarea completă şi mutarea cheilor utilizatorilor în custodia acestora nu elimină necesitatea sub-reţelei de securitate. Aceasta conţine de exemplu şi politici de acces la sub-reţeaua de date, care vor fi întotdeauna salvate în sub-reţeaua de securitate. Sub-reţeaua de securitate va fi administrată de beneficiarul principal al infrastructurii dar există şi posibilitatea ca aceasta să fie administrată de mai multe părţi, acest lucru fiind la îndemâna beneficiarului.

Sub-reţeaua de date, conţine efectiv nodurile cu date care fac scopul existentei produsului software. Crearea mai multor sub-reţele de date se face deoarece acestea pot fi configurate separat de către administratorul sistemului în aşa fel încât anumite sub-reţele de date să nu poată primi conexiuni decât de la anumite adrese IP. De asemenea, anumite sub-reţele pot fi configurate sa nu permită sincronizarea anumitor date in noduri din alte sub-reţele de date. De exemplu: se poate configura o anumită entitate (tabela), să nu se sincronizeze decât cu noduri din interiorul sub-reţelei în care se află.

Cu referire la Figura 1 anexată, sunt ilustrate următoarele caracteristici:

Element Grafic	**Descriere**
Dreptunghi rosu	Sub-reţea securitate
Dreptunghi verde	Sub-reţea de date
Săgeata roşie	Canal de comunicare al utilizatorilor în vederea autorizării şi autentificării
Săgeata albastra	Canal de comunicare şi sincronizare a datelor intre sub-reţele
Săgeata neagra	Canal de comunicare şi sincronizare a datelor intre noduri
Săgeata verde	API - interfaţa comunicare programatori
	Nod în sub-reţea
	Sisteme externe care se pot conecta cu HyperFiuse.
	Interfaţa administrare HyperFiuse prin intermediul căreia administratorul configurează sistemul

Cu referire la Figura 2 anexată, sunt ilustrate următoarele componente:

Componenţă nod	**Descriere şi scop**
Aplicaţie software	Componenta software care foloseşte platforma HyperFiuse
Interfaţă API	Interfaţa de comunicare, care permite programatorului sa interacţioneze cu platforma HyperFiuse. Interfaţa API prezintă conectori de comunicare care folosesc protocoalele HTTP(REST) şi TCP.
Interfaţă GraphQL	Interfaţa de interogare a datelor. Interfaţa API permite interacţiunea cu datele în sistem CRUD (create, read, update, delete). Pentru mecanisme de interogare mai complexa a datelor se foloseşte interfaţa GraphQl. Aceasta componenta este de asemenea dedicata programatorilor ca şi o unealta de accesare a datelor.
Motor procesare date	Motorul HyperFiuse. Toată funcţionalitate care face parte din produsul HyperFiuse este concentrata în aceasta componenta. Fiecare funcţionalitate va fi explicata detaliat

	în capitolele următoare.
Interfaţă adaptor blockchain	Interfaţa de conectare intre motor HyperFiuse şi motor blockchain. Aceasta componenta permite inter-schimbarea motorului de blockchain cu orice motor blockchain preferat de beneficiar si permite HyperFiuse sa fie o platforma agnostica în acest punct de vedere
Platforma/Motor blockchain	Motor de blockchain precum Hyperledger Fabric, Tendermint, Ethereum s.a.m.d
Interfaţă adaptor baza de date	Interfaţa de conectare intre motor HyperFiuse şi motor de baze de date. Aceasta componenta permite inter-schimbarea motorului de baza de date cu orice motor blockchain preferat de beneficiar şi permite HyperFiuse sa fie o platforma agnostica în acest punct de vedere
Baza de date	Motor de baza de date precum Oracle, MongoDb, MsSQL s.a.m.d.

3.3.1 Mecanism stocare date

Platforma conform prezentei invenţii (HyperFiuse) poate fi denumită şi un produs software conector a unui motor de blockchain (HyperLedger, Tendermint, Ethereum etc) cu un motor de management a bazelor de date (MongoDb, MsSql, Oracle etc), permiţând programatorilor sa comunice cu un singur sistem, care la rândul lui se va ocupa de stocarea datelor în ambele componente conectate (blockchain şi baza de date), precum şi cu sincronizarea datelor stocate pe toate nodurile de date care compun reţeaua.

Mecanismul de bază constă în preluarea informaţiilor care trebuiesc salvate în sistemul de stocare, calcularea unei amprente a informaţiei preluate folosind algoritmi de *hashing*, salvarea datelor preluate în motorul de baze de date, şi a amprentei, împreuna cu referinţa locaţiei datelor (tabela/colecţia din baza de date şi identificatorul unic al datelor în tabela/colecţie) în motorul de blockchain.

Primul pas al procedeului de stocare de date este preluarea datelor. Acest proces se face cu ajutorului interfeţei API, prin intermediul căreia aplicaţia software trimite informaţiile care se doresc a fi stocate în sistem *blockchain database*. După cum se vede şi în figura *Mecanism stocare date* de mai jos, aplicaţia software trimite un pachet de date de forma *{ "nume student":"Alin", "prenume student":"IFTEMI" }*. Formatul de date trimis difera în funcţie de entitatea/tabela în care se doreşte a fi stocata informaţia, structura acesteia fiind

definita în prealabil prin intermediul metodelor API expuse în componenta *Interfaţa API*, sau prin intermediul aplicaţiei de administrare a sistemului HyperFiuse.

Pasul al doilea consta în preluarea datelor, de către motorul HyperFiuse, pentru validare şi salvare în baza de date respectiv salvarea în blockchain (mecanismul de blocuri imutabile). Configurarea standard, de început, prevede ca fiecare structura de date care este trimisa pentru salvare sa fie stocata în baza de date, iar pe blockchain sa se salveze o referinţa la locaţia datelor în baza de date(Tabela/colecţie şi identificator unic înregistrare) precum şi o amprenta digitala a acestor informaţii.

Pentru exemplul prezentat în figura *Mecanism stocare date*, informaţia preluata de motorul de procesare de date HyperFiuse, şi anume *{ "nume student":"Alin", "prenume student":"IFTEMI" }*, este împărţita în doua seturi de date, cate una pentru fiecare componenta de stocare, baza de date respectiv blockchain (blocurile de date imutabile din motorul de blockchain).

In acest fel, forma informaţiei stocate in baza de date se prezintă in structura *{ "id":"XXX", "nume student":"Alin", "prenume student":"IFTEMI" }*. Câmpul "*id*" din structura prezentata se refera la identificatorul unic care s-a obţinut in urma salvării. In exemplul prezentat, valoare "XXX" poate reprezenta orice identificator unic asociat informaţiei salvate de către motorul de baze de date. Fiecare data trimisa către stocare este direcţionată către tabela/colecţia asociata acesteia in sistemul de gestionare a bazelor de date (motorul de baze de date).

După salvarea informaţiei in baza de date, se calculează amprenta digitală a înregistrării salvate folosind o metoda/funcţie de hash-ing (MD5, SHA256, SHA512). Sistemul poate folosi orice metodă de hash-ing agreată de administrator, selecţia acesteia făcându-se în baza unui parametru configurabil al HyperFiuse. Amprenta digitală, împreună cu numele tabelei din sistemul de gestiune a bazelor de date în care s-au salvat datele, şi cu identificatorul unic al înregistrării, se grupează într-un set nou de date de forma *{ "id":"YYY", "Tabela / colecţie":"Studenţi", "identificator unic":"XXX", "amprenta pachet date motor baza de date":"ZZZ" }* şi se trimit către salvarea in blockchain (motorul de blockchain); Câmpul "id" reprezintă identificatorul unic al tranzacţiei asociată de către motorul de blockchain, unde cu valoarea "YYY" poate reprezenta orice valoare dată de către un astfel de sistem. În acest mod, se creează o legătură între informaţiile salvate în baza de date şi informaţiile salvate în blockchain, punctul de legătură fiind numele tabelei în care s-au salvat datele şi identificatorul unic al înregistrării din tabel.

Structura tabelei, entităţii sau colecţiei în care se salvează informaţia este definită în prealabil de administratorul sistemului folosind Interfaţa API sau aplicaţia web de configurare a HyperFiuse. Structura de date folosita ca exemplu in acest document are ca scop prezentarea conceptului de stocare a datelor şi a conexiunii intre sistemele de blockchain cu sistemele de baze de date. În sistemul de producţie, aceste structuri pot fi diferite prin număr de câmpuri sau denumire câmpuri, variantele prezentate mai sus folosind doar pentru scop explicat şi educational. Câmpuri din motorul de blockchain precum: nodul unde a fost salvată înregistrarea, data/ora la care o înregistrare s-a făcut şi alte câmpuri, au fost omise întocmai a evidenţia mai simplu şi mai clar mecanismul de funcţionare a sistemului.

Procedeul descris este similar şi pentru ştergerea datelor. Operaţiunile de scriere la nivel de baza de date pot fi de trei feluri: inserare, modificare şi ştergere. Fiecare operaţiune este stocata in blockchain ca şi tranzacţie independenta alături de informaţiile de referinţa precum tabela, identificatorul înregistrării şi amprenta digitala. Practic, fiecare operaţiune de modificare a datelor este salvata ca o tranzacţie pe blockchain şi toate operaţiunile sunt păstrate integral pe toată durata de viata a aplicaţiei. La nivel de baza de date, operaţiunile pot modifica datele conform tipului de tranzacţie solicitata de aplicaţia software (programator) : creare, modificare sau ştergere. Tabela/colecţia din baza de date asociata structurii de date asupra căreia se fac modificări, păstrează in permanenta ultima versiune a datelor dar pentru fiecare tabela/colecţie există tabela/colecţie corespondentă care păstrează toate versiunile anterioare ale înregistrării tranzacţionate.

Acest mecanism este creat pentru a păstra un istoric imutabil al operaţiunilor (blockchain) dar şi versiunea integrala a datelor precum şi istoricul acestora. Există posibilitatea de a configura HyperFiuse să nu păstreze istoricul datelor modificate, sau ca acest istoric sa poate fi şters pentru anumite înregistrări. Aceasta funcţionalitate este utilă pentru îndeplinirea cerinţelor legale GDPR, iar parametrizarea acestor operaţiuni poate fi făcuta numai pentru tabelele/colecţiile selectate de administratorul sistemului şi la iniţiativa acestuia.

3.3.2 Mecanism distribuţie date

HyperFiuse se prezintă sub forma unei reţele de noduri, iar distribuţia/sincronizarea datelor este funcţionalitate principala pentru ca sistemul sa poate oferi scalabilitate, descentralizare şi rezistenta la modificarea datelor. Reţeaua de date HyperFiuse poate fi compusa din mai multe sub-reţele (vezi capitolul 3.2), iar fiecare sub-reţea de date conţine mai multe noduri. Fiecare nod, conţine o copie integrala a datelor din blockchain (blocuri imutabile de referinţe tranzacţii) precum şi o copie a bazei de date (vei capitolul 3.3). La salvarea unui

set de date, tipul de operaţiune, referinţa şi amprenta digitala sunt salvate in blockchain iar datele sunt direcţionate către baza de date unde se stochează in tabelele/colecţiile corespunzătoare (vezi capitolul 3.3.1).

Odată trimise datele către motorul de blockchain, acesta începe sa facă sincronizarea cu celelalte instante instalate pe nodurile de date, folosind algoritmii de consens care au fost configuraţi la instalare. Procesul de sincronizare al motorului de blockchain se ocupa şi de crearea de blocuri de tranzacţii (înregistrări), aceasta funcţionalitate fiind in totalitate specifica motorului de blockchain, iar eficienta sincronizării precum şi viteza de sincronizare fiind strict dependenta de configurările şi de motorul de blockchain ales de beneficiar. HyperFiuse integrează motoare de blockchain in funcţie de preferinţa beneficiarului şi nu intervine in mecanismele de sincronizare ale acestora.

Datele din blockchain se distribuie pe absolut toate nodurile din toate sub-reţelele de date, sub autoritatea motorului de blockchain, dar datele din baza de date sunt sincronizate de sistemul HyperFiuse. Fiecare nod din sub-reţeaua de date este actualizat in blockchain cu blocurile cele mai recent validate de motorul de blockchain, iar odată sincronizat un nou bloc, sistemul HyperFiuse procesează fiecare tranzacţie din bloc şi pentru fiecare dintre acestea iniţiază o cerere de obţinere a datelor modificate către nodul emitent in care a fost creata tranzacţia. Odată primite datele, acestea sunt modificate şi in baza de date a nodului recipient conform tipului de tranzacţie specificat in blockchain. Modificarea acestora se refera la oricare dintre operaţiunile de alterare a datelor: creare, modificare şi ştergere precum şi la crearea istoricului înregistrării tranzacţionate.

Timpul, respectiv viteza de sincronizare a datelor din blockchain este direct dependenta de motorul de blockchain folosit (HyperLedger, Tendermint, Ethereum etc), fiind la latitudinea beneficiarului alegerea unui motor de blockchain convenabil in funcţie de tipul de activitate al produsului software deservit de HyperFiuse. În general, motoarele de blockchain pentru aplicaţii industriale sunt optimizate pentru performanta, dar cu toate acestea se pot diferenţia prin viteza sau algoritmul de sincronizare, configurările şi modalitatea de stocare a datelor din chain-ul imutabil sau prin funcţionalitate comerciala. Hyperfiuse este independent de motorul de blockchain, mai multe distribuţii fiind disponibile pentru ca beneficiarul sa îşi aleagă distribuţia cea mai convenabila, atât d.p.d.v. motor de blockchain cat şi motor de baze de date. Mai multe detalii despre independenta fata de motorul de blockchain in capitolul 3.3.4.

3.3.3 Mecanism citire date

Accesul la date oferit de HyperFiuse se face prin doua moduri: acces direct la baza de date şi acces prin Interfaţa API.

Accesul direct la baza de date se refera la publicarea detaliilor de conectare directa la motorul de baze de date. Acest tip de conectare este folosit in situaţia in care aplicaţia software foloseşte interogaţii complexe bazate pe T-SQL sau se doresc citiri rapide din baza de date, citiri cu timp de răspuns asumat de motorul de baza de date. Aplicaţia software se conectează direct la baza de date pentru a efectua citiri şi in cazul unei migrări rapide, de la o infrastructura bazata pe sisteme de gestiune a bazelor de date clasice la o infrastructura bazata pe HyperFiuse blockchain database, când mai pot exista situaţii când nu s-a finalizat modificarea codului sursa pentru a efectua citirea prin HyperFiuse, acel lucru urmând a se realiza mai târziu.

Există şi dezavantaje pentru acest tip de citire, şi anume că nu se pot obţine automat datele care au fost criptate în sistem, decriptarea va trebui sa fie făcuta manual de către programator (vezi capitolul 3.3.8). De asemenea, datele nu vor fi filtrate după drepturile de acces ale utilizatorilor asupra acestora (vezi capitolul 3.3.11). La operaţiunile de citire prin Interfaţa API, decriptarea datelor se face automat şi în funcţie de drepturile utilizatorului asupra datelor.

Metoda recomandata este accesarea înregistrărilor prin Interfaţa API. HyperFiuse va returna informaţiile solicitate de programator din Aplicaţia Software, iar acestea ii vor fi livrate ţinând cont de toate configurările şi filtrele de funcţionalitate HyperFiuse. Concret, citirea prin Interfaţa Api va returna datele în format citibil (decriptate), va filtra datele în funcţie de drepturile pe care utilizatorul care efectuează citirea le are asupra acestora, va efectua verificări de integritate asupra datelor şi va permite citirea cu opţiunea de reconstruire a datelor în cazul în care au fost identificate modificări/ştergeri neautorizate asupra acestora. De asemenea, prin Interfaţa API, se pot efectua foarte uşor şi eficient operaţii de citire a versiunilor anterioare a unei înregistrări. Toate aceste beneficii sunt descrise în capitolele următoare şi nu sunt disponibile la citirea informaţiilor direct din sistemul de gestiune a bazei de date.

Viteza de citire a datelor prin Interfaţa Api, este este aceiaşi cu viteza de citire direct din baza de date, la care se adaugă (unde este cazul) timpul de decriptare, timpul de verificare a integrităţii datelor, timpul de reconstrucţie a unei înregistrări şi de filtrare a datelor asupra cărora utilizatorul care face citirea are acces. Toate aceste operaţiuni adiţionale aduc o

întârziere minora unei operaţiuni de citire, întârziere care ar fi apărut oricum în momentul în care programatorul ar fi decis sa implementeze singur aceste funcţionalităţi la nivel de aplicaţie software. De asemenea, toate aceste beneficii oferite de HyperFiuse care se adaugă la timpul de citire sunt opţionale şi la latitudinea administratorului de sistem şi a programatorului.

3.3.4 Independenta faţă de motorul de blockchain

HyperFiuse foloseşte componenta de blockchain, motorul de blockchain, pentru a sincroniza blocurile de date cu tranzacţii, referinţe la datele operate şi amprenta digitala a acestora, precum şi pentru a păstra un istoric imutabil al tuturor operaţiunilor asupra datelor. Întregul set de blocuri de date al motorului de blockchain este folosit de HyperFiuse pentru reconstrucţia şi sincronizarea datelor din motorul de baze de date.

Tehnologia blockchain este la început de drum, nu este încă diferenţiat un motor de blockchain in detrimentul altuia şi se experimentează foarte mult pe partea de algoritmi de consens şi viteza de sincronizare. Exista deja multi producători de software care au evoluat considerabil in ceea ce priveşte cercetarea şi dezvoltarea in domeniul motoarelor de blockchain, şi care au scos pe piaţa produse fiabile, dar tehnologia este încă la început de drum.

Datorită şi faptului că există mai mulţi producători de motoare de blockchain, iar nevoile clienţilor/beneficiarilor se pot raporta şi la acest lucru, HyperFiuse se prezintă ca o platforma independenta de aceasta componenta. Asta înseamnă ca HyperFiuse poate funcţiona şi cu HyperLedger Fabric (motor de blockchain), şi cu Tendermint şi cu oricare motor de blockchain îşi doreşte beneficiarul aplicaţiei. HyperFiuse va fi distribuit beneficiarului cu o combinaţie de motor blockchain şi motor baza de date la solicitarea acestuia sau ii va fi distribuit în forma descărcabilă de pe site-ul companiei, fiind la latitudinea dezvoltatorului sa creeze o diversitate cat mai mare de distribuţii pentru a acoperi nevoile tuturor beneficiarilor.

3.3.5 Independenta fata de motorul de baze de date

Fiecare nod HyperFiuse conţine şi un sistem de gestiune a bazelor de date, motorul de baze de date. Aceasta componenta este folosita pentru a stoca în întregime înregistrările şi datele salvate in sistem precum şi pentru a păstra versiunile anterioare ale acestora.

Este bine cunoscută dependenta anumitor beneficiari de anumiţi producători/furnizori de sisteme de gestiune a bazelor de date, şi cum HyperFiuse a fost creat sa ofere şi posibilitatea migrării aplicaţiilor software existente in tehnologia blockchain, a devenit

imperios necesar ca aceasta componenta, motorul de baze de date, sa poată fi schimbata in funcţie de preferinţa beneficiarului.

Datorită şi faptului că există mai mulţi producători de motoare de baze de date, iar nevoile clienţilor/beneficiarilor se pot raporta şi la acest lucru, HyperFiuse se prezintă ca o platforma independenta de aceasta componenta. Asta înseamnă ca HyperFiuse poate funcţiona şi cu Oracle (motor de baze de date), şi cu MongoDb şi cu oricare motor de baze de date îşi doreşte beneficiarul aplicaţiei. HyperFiuse va fi distribuit beneficiarului cu o combinaţie de motor blockchain şi motor baza de date la solicitarea acestuia sau ii va fi distribuit în forma descărcabilă de pe site-ul companiei, fiind la latitudinea dezvoltatorului sa creeze o diversitate cat mai mare de distribuţii pentru a acoperi nevoile tuturor beneficiarilor.

3.3.6 Polițe de sincronizare a datelor

Conform descrierii din capitolul 3.2, sub-reţelele de date pot fi configurate pentru a limita sincronizarea datelor din anumite tabele/colecţii la nodurile din cadrul sub-reţelei. Polițele de sincronizare a datelor se aplica la nivel de sub-rețea, unde pentru fiecare sub-rețea de date se poate specifica ce tabela/colecţie de date sa nu se sincronizeze cu noduri din afara sub-reţelei.

Polițele de sincronizare a datelor permit şi restricţionarea cererilor de furnizare date adresate nodurilor din sub-rețea prin intermediul Interfeţei API. Aceasta restricţionare se face la nivel de adresa IP, şi odată definita o astfel de poliţa de restricţionare, nu se mai poate comunica cu Interfaţa API decât de către Aplicaţiile Software care sunt într-o marja de IP conforma cu configurarea sub-reţelei de date.

O sub-rețea de date care nu are definita nici o poliţa de sincronizare a datelor se numeşte OPEN NETWORK, iar o sub-rețea de date care are definita cel puţin o regula într-o poliţa de sincronizare a datelor se numeşte RESTRICTED NETWORK.

Un exemplu de scenariu in care o poliţă restrictiva la sincronizarea datelor ar avea sens, este pentru cazul în care China nu permite transferul documentelor personale stocate în format electronic sa părăsească graniţele fizice ale tarii. Dacă un beneficiar din China ar desfăşura activităţi în China şi in Korea, iar in Korea nu ar exista astfel de restricţii, atunci reţeaua de date HyperFiuse ar putea fi împărţită cu doua sub-reţele de date, una in China şi una in Korea. Pentru sub-reţeaua din China vom crea o poliţa de sincronizare a datelor care va include tabela "Documente" ca tabela/colecţie care nu are voie sa sincronizeze date în alte sub-reţele şi vom adăuga in poliţa de sincronizare şi o regula de blocare a cererilor de acces de la orice IP care nu este din China. (este important de ştiut ca descărcarea în format electronic a

unui document accesat dintr-un browser din Europa, este considerat de asemenea o breşă care rezulta într-o ilegalitate; de aceea este necesara şi limitarea accesului la Interfaţa API pe nodurile din sub-reţeaua China). Restricţiile in ceea ce priveşte sincronizarea datelor se pot aplica atât către datele care pleacă dintr-o anumita sub-reţea cat şi la datele vin din alte sub-reţele, de asemenea aceste restricţii se aplica la nivel de tabela/colecţie din sub-reţea de date.

În figurile 4-6 anexate sunt definite următoarele scenarii:

- *Fig. 4 Reţea deschisă (Open Network)* - exemplu de reţea in care nu s-a definit nici un fel de restricţii la nivel de tabela/colecţie "Documente". Este un scenariu clasic, in care nu se definesc restricţii dar se definesc sub-reţele pentru a păstra opţiunea deschisa in viitor. Deşi nu exista nici o diferenţa intre a avea o reţea neîmpărţită in sub-reţele şi a împarţi reţeaua in doua sub-reţele de tip OPEN_NETWORK (fără restricţii) , este indicata împărţirea in sub-reţele din punct de vedere logic, pentru a eventuale configurări sau limitări care pot apărea in producţie care necesita implementări rapide.
- *Fig. 5 Reţea restricţionată 1 (Restricted Network 1)* - exemplu de reţea in care s-a definit pe tabela Documente din sub-reţeaua China o regula de restricţie pentru ca datele introduse prin nodurile din sub-reţeaua China sa nu se sincronizeze decât in sub-reţeaua China şi sa nu permită sincronizarea cu noduri din sub-reţele externe. Regula permite însa ca tabela sa primească înregistrări din alte sub-reţele de date
- *Fig. 6. Reţea restricţionată 2 (Restricted Network 2)* - exemplu de reţea în care s-a definit pe tabela Documente din sub-reţeaua China o regula de restricţie pentru ca datele introduse prin nodurile din sub-reţeaua China sa nu se sincronizeze decât in sub-reţeaua China şi sa nu permită sincronizarea cu noduri din sub-reţele externe. Regula NU permite nici ca tabela sa primească înregistrări din alte sub-reţele de date. Acelaşi efect s-ar fi obţinut dacă ambele sub-reţele ar fi restricţionat tabela "Documente" numai la trimiterea de date. In ambele variante nu se regăsesc diferenţe la nivelul in care sunt distribuite datele dar modalitate de sincronizare ar fi fost diferită dacă am mai introduce o alta sub-reţea..

3.3.7 Polițe de stocare a datelor

HyperFiuse este o reţea distribuita de noduri, unde fiecare nod retine o copie integrala, o replica, a bazei de date. Multiplicarea nedeterminata a nodurilor, respectiv a întregului set de date din baza de date, poate crea probleme de logica şi de infrastructura. Nu întotdeauna are sens ca un nod, în mentenanţa unei parţi, sa conţină toate datele din sistem la fel cum nu are sens sa se întreţină prea multe noduri cu toate datele, din perspective tehnica a consumului de

resurse hardware. Pentru a rezolva aceasta problema, HyperFiuse permite configurarea unui nod în asa fel încât acesta sa nu sincronizeze toate datele, şi sa stocheze numai datele care au fost salvate folosind Interfaţa API instalata pe nodul în cauză.

În acest mod, un nod în reţeaua HyperFiuse poate fi un nod care stochează toate datele din sistem, iar un astfel de nod se numeşte FULL NODE. Aceste noduri de obicei se găzduiesc în sediul central sau într-un centru de date care aparţine beneficiarului, şi pot servi ca sursa pentru recuperare de date sau ca un instrument de back-up în timp real. Scopul fiind de a proteja nodurile de business de la parteneri sau filiale, de a întreţine volume mari de date cu hardware scump. Funcţionalitatea este opţională, fiind posibil fără probleme, în funcţie de scop, ca toate nodurile dintr-o reţea sau sub-reţea sa funcţioneze ca un FULL NODE.

Un nod poate funcţiona fără a sincroniza toate datele care se salvează pe toată reţeaua, păstrând doar datele care au fost introduse folosind Interfaţa API a nodului respectiv. Aceste noduri nu vor păstra în baza de date decât datele care au fost introduse prin Interfaţa API a nodului în cauza. Acest procedeu nu restricţionează accesul la alte date introduse prin alte noduri, acestea fiind replicate în momentul în care sunt solicitate de utilizator. Odată copiate, în urma unei solicitări, datele aduse vor rămâne pe nodul parţial pentru totdeauna. Scopul acestei funcţionalităţi este de a oferi posibilitatea unui software sa comunice cu un nod de date întreţinut într-un mediu de afaceri, precum o filiala sau un birou mic, fără a fi nevoie de o infrastructura hardware complexa instalata la fata locului, dar şi pentru a evita sincronizarea anumitor date care aparţin altor parţi pe baza de date locala. HyperFiuse poate deservi ca soluţie de stocare de date, aplicaţii software care sunt operate de mai multi terţi care sunt foarte interesaţi ca datele pe care aceştia le introduc în sistem sa nu fie expuse celorlalte parţi care operează sistemul. Chiar dacă aceste date pot fi criptate, se induce o siguranţa sporita în momentul în care, deşi criptate, datele nu sunt deloc accesibile altor parţi. Aceste noduri poartă denumirea de PARTIAL NODE.

Un nod PRIVATE, este un alt tip de nod, care nu numai că nu copiază local datele care au fost introduse prin interfaţa API a altor noduri, dar nici nu permite ca datele introduse prin Interfaţa API proprie să fie sincronizate pe alte noduri. În figura 7 se poate observa o reprezentare a unei reţele cu un nod FULL NODE şi 2 noduri PARTIAL NODE

3.3.8 Criptarea datelor

Criptarea datelor este folosita în HyperFiuse pentru a îmbunătăţi protecţia şi securitatea datelor stocate în componenta de baza de date, şi a oferi posibilitatea beneficiarului sa ascundă anumite informaţii senzitive în cazul unor extrageri de date neautorizate dar şi pentru a oferi

utilizatorilor sistemului posibilitatea de a deveni proprietari ai datelor nu numai la nivel conceptual ci şi la nivel fizic, aceste date aflate în proprietatea utilizatorului fiind criptate cu cheia privata a acestuia.

Mecanismele de criptare sunt bazate pe algoritmi de criptare asimetrici, care folosesc un cuplu de chei format din cheie publica pentru criptare şi cheie privata pentru decriptare. Mecanismul de funcţionare al acestui mod de criptare nu este explicat in acest document, dar pentru mai multe informaţii se poate accesa acest link https://en.wikipedia.org/wiki/Public-key_cryptography

HyperFiuse foloseşte doua entităţi care pot cripta/decripta informaţii în sistem, mai exact administratorul/beneficiarul sistemului şi utilizatorul/utilizatorii care adaugă informaţii în sistem. Mecanismul de criptare este cu precădere destinat pentru a proteja informaţiile adăugate de utilizatori dar administratorul are funcţia de criptare/decriptare pentru a putea interveni în recuperarea unor date care au fost criptate de un utilizator, iar acesta din urma şi-a pierdut cheia privata de decriptare sau efectiv nu este disponibil pentru a decripta la o cerere de acces a informaţiilor iniţiata de către un alt utilizator. Mecanismul de acces la informaţii, bazat pe cererile utilizatorilor este descris în detaliu în capitolul 3.3.11

Cuplul de chei format din cheie publica şi cheie privata, care aparţine administratorului se numeşte cuplu de chei MASTER. Cheia privata MASTER este împărţita în mai multe bucăţi, astfel încât aceasta sa nu poată fi reconstruita decât dacă sunt prezenţi mai multi administratori în procesul de decriptare urgenta. Scopul acestei funcţionalităţi este sa se prezinte încredere în beneficiarul sistemului, care deşi poate decripta informaţii în regim de urgenta, acest procedeu sa nu se poată face decât într-un mediu controlat, cu mai multi administratori de fata şi cu toţi paşii efectuaţi din aceasta operaţiune logaţi într-un sistem imutabil de loguri. Cheia privata este împărţita in N bucăţi (configurabil) fiind nevoie de K bucăţi pentru a reconstrui cheia (unde K <= N). Algoritmul folosit pentru a împarţi cheia privata se numeşte "Shamir's Secret Sharing" şi explicarea lui nu face scopul acestui document dar mai multe detalii despre acest algoritm se pot vizualiza la acest link https://en.wikipedia.org/wiki/Shamir%27s_Secret_Sharing Fiecare bucata din cheia privata MASTER care este în guvernarea unui administrator este criptata la rândul ei cu cuplul de chei asimetrice care ii aparţine acestuia, pentru a preveni un administrator sa vadă în sistem bucata din cheia privata MASTER care ii aparţine altui administrator. Configurabil, se poate opta ca cheia asimetrica a unui administrator sa fie o cheie hardware, un dispozitiv extern, sau ca aceasta sa fie stocata în sub-reţeaua de securitate şi sa se genereze în baza numelui de utilizator şi a parolei administratorului.

Utilizatorilor HyperFiuse li se crează un cuplu de chei asimetrice, iar aceştia vor folosi cheia publica pentru criptare şi cheia privata pentru decriptare. Cheia publica este disponibila şi vizibila tuturor utilizatorilor din sistem, dar cheia privata este secreta şi aparţine fiecărui utilizator în parte. Cheia privata a unui utilizator este protejata de HyperFiuse prin 3 mecanisme diferite şi anume: dispozitiv hardware extern care stochează cheia privata şi este în posesia utilizatorului, păstrarea cheii private de către utilizator în format sir de caractere digital sau varianta mai puţin securizata, în care cheile sunt stocate în sub-reţeaua de securitate şi gestionate de sistem. A treia varianta în care cheile private sunt gestionate de sistem, sunt destinate produselor software care nu au utilizatori pregătiţi sa îşi protejeze o cheie privată şi pun accent pe alte funcţionalităţi oferite de HyperFiuse sau acorda un gram de încredere superior beneficiarilor platformei, din perspectiva faptului ca aceştia pot oferi protecţia necesara a acestor chei în relaţia cu alte parţi.

Criptarea informaţiilor care se salvează în baza de date se face la solicitarea utilizatorilor de sistem. Aceştia pot configura sistemul ca anumite înregistrări sa fie stocate în baza de date criptat. Exista şi posibilitatea ca anumite câmpuri dintr-o înregistrare (eg. doar câmpul nume şi prenume dintr-o tabelă cu câmpurile nume, prenume şi data naşterii) sa fie criptate, restul dintre acestea sa fie stocate în clar, în forma în care au fost introduse de utilizator. Alegerea câmpurilor dintr-o entitate (tabela) care sa fie stocate criptat sta la latitudinea beneficiarului, şi este opţiunea acestuia sa gestioneze volumul de informaţii care se criptează în raport cu performanta care se aşteaptă de la sistem, având în vedere faptul ca este binecunoscuta situaţia în care procesul de criptare şi decriptare poate reduce performanta de acces la informaţie.

Informaţia care se criptează înainte de stocare, este criptata cu doua chei publice, cheia utilizatorului care introduce datele şi cheia MASTER care este folosita la decriptare în caz de urgenta. Sistemul poate fi configurat sa nu se folosească o cheie MASTER, şi recuperarea datelor sa nu se facă decât pe baza cheii private a utilizatorului, dar acest lucru se va face în cunoştinţa de cauza cu menţiunea ca dacă un utilizator îşi pierde cheia privata, este posibil ca datele criptate sa nu mai poată fi recuperate niciodată.

Programatorul care interacţionează cu HyperFiuse prin Interfaţa API nu se va îngrijora niciodată de procesul de criptare/decriptare, cheile asociate procesului fiind generate din timp, iar el va trimite întotdeauna informaţiile în formatul original necriptat şi le va extrage din sistem în acelaşi format. Procesul de criptare, stocare criptata şi decriptare este efectuat de HyperFiuse in mecanismele interioare, programatorii fiind scutiţi de efortul de a procesa aceste operaţiuni.

3.3.9 Decriptarea datelor

Decriptarea datelor in HyperFiuse are loc în urma unei solicitări de vizualizare a datelor din partea unui utilizator, în contextul în care datele au fost stocate criptat în baza de date. Aceasta decriptare se face cu cheia privata a utilizatorului care a introdus datele şi care este proprietarul acelor date. Decriptarea mai poate fi făcuta şi de administratorul sistemului cu cheia MASTER în cazurile speciale când utilizatorul a pierdut cheia privata sau când utilizatorul nu poate răspunde la o cerere urgenta de acordare a permisiunii asupra datelor (Ex: datele criptate sunt date medicale de importanta vitala şi utilizatorul este chiar posesorul cheii private în postura de a nu putea oferi acces la acestea)

Decriptarea datelor mai este invocată şi în cazul în care se solicita acces în baza unei cereri formulate de un alt utilizator. Proprietatea datelor nu restricţionează total accesul la acestea, datele deţinute pot fi chiar adăugate in sistem ca date publice sau ca date private. Cele publice sunt accesate de toţi utilizatorii iar cele private nu pot fi accesate de nici un alt utilizator decât de utilizatorul care le-a adăugat. Dar datele pot fi introduse şi cu caracteristica PERMISSIONED care permite celorlalţi utilizatori sa vadă ca datele exista în sistem dar aceştia trebuie sa solicite acces. In acest caz, dacă utilizatorul proprietar de date doreşte sa ofere acces, atunci se mai efectuează o decriptare a datelor cu cheia proprietarului de date iar datele sunt făcute disponibile pentru utilizatorul căruia i s-a acordat privilegii de citire.

Mecanismul de criptare şi decriptare a informaţiilor introduse de un utilizator sunt opţionale, şi activarea acestora este făcută de administratorul de sistem, de designerul de infrastructura de date în funcţie de logica produsului software şi aşteptările funcţionale ale produsului software care foloseşte platforma HyperFiuse.

3.3.10 Semnarea datelor

O semnătură digitală este o schemă matematică folosită pentru a demonstra autenticitatea mesajelor sau documentelor digitale. O semnătură digitală validă oferă destinatarului o bază solidă pentru a crede că mesajul a fost creat de către un expeditor cunoscut (autentificare), să fie sigur că expeditorul nu poate nega că a trimis mesajul (non-repudiere), şi că mesajul nu a fost modificat pe parcurs (integritate).

Semnăturile digitale sunt un element standard al celor mai multe suite de protocoale criptografice, şi sunt frecvent utilizate pentru distribuirea de software, tranzacţii financiare, software de gestiunea contractelor, precum şi în alte cazuri în care este important să se

detecteze falsificarea sau manipularea datelor. Mai multe informaţii despre semnătura digitala se găsesc la acest link:

https://ro.wikipedia.org/wiki/Semn%C4%83tur%C4%83_digital%C4%83

HyperFiuse utilizează semnătura digitala pentru a semna tranzacţiile care sunt stocate pe lanţul imutabil în blockchain. Fiecare tranzacţie este de fapt o operaţiune de scriere/modificare a unei înregistrări în baza de date, dar detaliile operaţiunii care sunt salvate în blockchain (vezi detalii în capitolul 3.3.1) sunt semnate digital. In blockchain se salvează tranzacţiile care sunt grupate în blocuri de tranzacţii. Fiecare tranzacţie din bloc este semnata digital cu cheile utilizatorului care a adăugat datele în sistem, iar fiecare bloc este semnat digital cu cheile nodului care a confirmat blocul de tranzacţii.

3.3.11 Proprietatea datelor şi accesul la date

Fiecare utilizator HyperFiuse este proprietarul informaţiilor pe care acesta le introduce în sistem. Acest lucru se confirma prin semnături digitale (vezi capitolul 3.3.10) şi prin criptarea informaţiilor introduse. Exista într-adevăr şi posibilitatea de a nu cripta informaţia, mai ales ca exista un timp preţios pierdut la decriptare, dar este la latitudinea administratorului de sistem sa decidă ce informaţie este sensibila şi necesita criptare pentru a proteja accesul la aceasta şi ce informaţie poate fi stocata necriptat fiind suficienta doar semnătura digitala pentru a demonstra proprietatea informaţiei.

Criptarea informaţiei stocate în baza de date, este mecanismul esenţial în protejarea informaţiilor unui utilizator. Mecanismul de criptare/decriptare explicat în capitolul 3.3.8 şi 3.3.9, se aplica având ca scop protecţia datelor în baza de date, date care pot fi replicate pe noduri aflata în gestiune unor terţi, dar şi pentru a proteja accesul la date fata de alţi utilizatori. Înregistrările stocate în sistem cu eticheta PERMISSIONED (vezi capitolul 3.3.9) sunt disponibile către alţi utilizatori, care pot vedea ca informaţia exista, dar pentru a putea-o accesa este nevoie de un mecanism de cerere de permisiune de acces, permisiune acordata în final de utilizatorul care este proprietar al informaţiei. Înregistrările stocate cu eticheta PRIVATE (vezi capitolul 3.3.9) nu vor putea fi niciodată accesate prin intermediul mecanismul de cerere de acces, dar un administrator va putea sa le vizualizeze direct în baza de date, iar acele informaţii ar trebui sa fie stocate criptat dacă acestea sunt informaţii sensibile.

Proprietatea datelor unui utilizator impune acestuia sa participe în procesul de cerere de acces la date (pentru înregistrările etichetate cu PERMISSIONED) şi este utilă ca datele sa fie stocate criptat în cazul în care informaţiile se doresc a fi distribuite către terţi în baza unei

cereri de acces. Asta nu înseamna ca datele pot fi distribuite şi necriptat, acestea nu vor fi disponibile prin Interfaţa API, dar vor putea fi vizibile unui administrator de date direct in baza de date. Mecanismele de configurare a câmpurilor, dintr-o colecţie sau entitate, care stochează informaţia criptat sunt gestionate de administratorul de sistem. Arhitectul sistemului de date va decide dacă o înregistrare este stocata criptat integral în baza de date sau doar anumite câmpuri din acea înregistrare vor stoca informaţia criptat. Eticheta de PERMISSIONED este transmisa de către programatorul care trimite informaţiile în sistem, şi fiecare înregistrare trimisa către salvare poate avea eticheta PERMISSIONED, PRIVATE sau PUBLIC. Sistemul software care comunica cu HyperFiuse, în baza regulilor decise de programator, poate trimite informaţiile cu una din cele 3 etichete indiferent de cum este configurata tabela şi/sau cate câmpuri sunt criptate sau nu.

În contextul în care o înregistrare este criptata în totalitate, se poate acorda acces numai la anumite câmpuri din aceasta. Utilizatorul care solicita acces la o înregistrare PERMISSIONED, cunoaşte structura tabelei în care informaţia este criptata, şi poate solicita accesa la întreaga înregistrare sau numai la câteva câmpuri din aceasta. De asemenea, proprietarul de date, poate acorda acces la toate câmpurile la care i s-a cerut acces sau poate decide căror câmpuri din cele solicitate, sa le acorde permisiune de citire. De asemenea, permisiunea de acces, se acorda pe o perioada de timp limitata, perioada de timp decisa de proprietarul datelor, iar la expirarea acestei perioade de timp informaţia nu mai poate fi accesata, şi este nevoie de o noua cerere de acces pentru a obţine din nou privilegiile de citire.

În tot acest proces de cerere şi acordare de acces la date etichetate cu PERMISSIONED, nu se face transfer de cheie privata de decriptare, informaţia fiind decriptata cu cheia proprietarului de date şi apoi aceasta este transmisa către utilizatorul care a solicitat acces, în forma clara necriptata prin Interfaţa API. Proprietarul datelor poate refuza o cerere de acces la date.

3.3.12 Vizibilitatea datelor în blockchain

Orice utilizator are acces la datele din HyperFiuse prin Interfaţa API. Interfaţa API permite unui utilizator sa interacţioneze cu sistemul pentru a adaugă, modifica şi şterge informaţii ca în orice sistem de gestiune a bazelor de date tradiţional. Operaţiunea de citire se face de asemenea prin intermediul Interfeţei API, iar citirea informaţiilor dintr-o tabela returnează calup de date la care utilizatorul care efectuează operaţia de citire are acces.

Operaţiunea de listare a informaţiilor dintr-o tabela, se poate efectua cu parametru de filtrare a informaţiilor la care un utilizator are acces:

- Listare informații cu parametru ALL ACCESSIBLE: se returnează toate informațiile introduse cu eticheta PUBLIC, toate informațiile introduse cu parametru PERMISSIONED care aparțin utilizatorului care face citirea sau altor utilizatori care au acordat acces la date utilizatorului care face citirea și toate informațiile introduse cu eticheta PRIVATE de către utilizatorul care face citire.
- Listare informații cu parametru ALL: se returnează toate informațiile la care un utilizator are acces (aceleași descrise mai sus la listarea cu parametru ALL ACCESSIBLE) la care se adaugă și toate informațiile introduse cu parametru PERMISSIONED de alți utilizatori și la care nu a fost acordata permisiunea de citire. Aceste informații vor fi returnate doar cu identificatorul unic al înregistrării și identificatorul utilizatorului care este proprietarul datelor, fără a dezvălui conținutul înregistrării, iar identificatorul unic va fi folosit de utilizatorul care a efectuat citirea pentru o solicitare de acces la date.
- Listare informații cu parametru ALL FORBIDDEN: se returnează toate informațiile introduse cu parametru PERMISSIONED de alți utilizatori și la care nu a fost acordata permisiunea de citire. Aceste informații vor fi returnate doar cu identificatorul unic al înregistrării și identificatorul utilizatorului care este proprietarul datelor, fără a dezvălui conținutul înregistrării, iar identificatorul unic va fi folosit de utilizatorul care a efectuat citirea pentru o solicitare de acces la date.

Indiferent ca informațiile sunt sau nu sunt criptate în baza de date, operațiunea de citire va returna lista înregistrărilor solicitate în forma clara, necriptata, procesul de decriptare fiind efectuat automat (în baza permisiunilor) de sistemul HyperFiuse, înainte ca acestea sa fie livrate către utilizatorul care efectuează operațiunea de citire.

3.3.13 Citirea cu confirmarea a integrității datelor

Operațiunea citirii individuale a unei înregistrări, în baza identificatorului unic al acesteia, se poate efectua și cu cerere de verificare a integrității datelor, pentru a se identifica dacă informațiile returnate de operațiunea de citire nu au suferit modificări neautorizate la nivel de baza de date. Operațiunea de verificare a integrității este opțională, iar în cazul în care se optează pentru aceasta, rezultatul returnat în urma solicitării înregistrării către Interfața API, va conține un câmp care are valoarea logica ADEVARAT sau FALS (true / false) care va indica exact dacă informația a fost alterata sau nu. Este la latitudinea programatorului sau a sistemului software care solicita informația, sa decidă ce acțiuni se pot deduce după citire, în cazul în care o informație a fost identificata ca modificata neautorizat. Utilizatorul care a

efectuat operaţia de citire poate contacta administratorul de date şi solicita investigaţii mai ample sau poate solicita sistemului sa reconstruiască informaţia şi sa o readucă la forma originala (vezi capitolul 3.3.14)

Verificarea informaţiilor se face prin calcularea unui hash al datelor citite din baza de date, şi compararea acestuia cu hash-ul corespondent înregistrării din blockchain. In blockchain, hash-ul original fiind calculat şi stocat la momentul introducerii datelor în sistem. Dacă cele doua hash-uri sunt diferite înseamnă ca informaţia a fost modificata în baza de date. Se pleacă de la premiza ca informaţia din blockchain nu poate fi modificata, iar falsificarea acesteia presupune reconstrucţia întregului lanţ imutabil de date pe toate nodurile.

Indiferent ca informaţiile sunt sau nu sunt criptate în baza de date, operaţiunea de citire va returna înregistrarea solicitate în forma clara, necriptata, procesul de decriptare fiind efectuat automat (în baza permisiunilor) de sistemul HyperFiuse, înainte ca aceasta sa fie livrata către utilizatorul care efectuează operaţiunea de citire.

3.3.14 Reconstrucţia datelor alterate

Operaţiunea citirii individuale a unei înregistrări, în baza identificatorului unic al acesteia, se poate efectua şi cu cerere de reconstrucţie a înregistrării. Este la latitudinea programatorului dacă decide sa reconstruiască o informaţie în cazul în care aceasta a fost identificata ca alterata fata de forma originala. Programatorul sau sistemul software care operează citirea, poate solicita reconstrucţia înregistrării independent de eventuale modificări ale acesteia. Opţiunea de reconstrucţie a datelor poate fi invocata la orice citire a informatei.

Mecanismul de reconstrucţie, odată invocat, va cauta o varianta nealterata a înregistrării pe nodul de date unde informaţia a fost adăugata sau pe un nod de tip FULL NODE din reţea. Mecanismul de reconstrucţie este creat pentru a oferi variante rapide pentru recuperarea datelor şi pentru a oferi garanţie superioara a imutabilităţii datelor şi rezistentei sporite ale datelor la modificări neautorizate.

Indiferent ca informaţiile sunt sau nu sunt criptate în baza de date, operaţiunea de citire va returna înregistrarea solicitate în forma clara, necriptata, procesul de decriptare fiind efectuat automat (în baza permisiunilor) de sistemul HyperFiuse, înainte ca aceasta sa fie livrata către utilizatorul care efectuează operaţiunea de citire.

3.3.15 Mecanismul de versionare a datelor

Toate datele stocate în HyperFiuse, sunt păstrate împreuna cu tot istoricul modificărilor acestora. Orice tabela/colecţie creata în HyperFiuse are un corespondent în baza de date

indiferent ce motor de baze de date este folosit. Pe lângă aceasta tabela, se crează o tabela istoric care va stoca toate versiunile anterioare ale unei înregistrări. Mai exact, în tabela din baza de date vom avea întotdeauna ultima versiune a unei înregistrări iar în tabela asociata de istoric vom avea toate versiunile anterioare ale înregistrării.

Versiunile istorice ale unei înregistrări, sunt şi ele imutabile, adică pot fi verificate cu hash-ul original stocat în blockchain, iar mecanismul de verificare a integrităţii (vezi capitolul 3.3.13) şi mecanismul de reconstrucţie de date (vezi capitolul 3.3.14) se aplica şi în cazul datelor istorice.

Operaţiunea citirii individuale a unei înregistrări, în baza identificatorului unic al acesteia, se poate efectua şi cu cerere de vizualizare a unei versiuni istorice a unei înregistrări. Orice citire, returnează odată cu datele solicitate şi versiunea maxima a înregistrării (număr întreg de la 1 la n unde n este ultima versiune). In baza acestui număr de versiune maxima, programatorul poate solicita orice versiune din intervalul 1-n a înregistrării respective în asa fel încât HyperFiuse va şti ce fel de informaţie se solicita şi va returna datele din tabela principala sau din tabela istoric.

3.3.16 Mecanismul de recuperare a datelor (back-up in timp real)

HyperFiuse înlocuieşte mecanismul de back-up clasic prin copiere integrala a bazei de date (tip snapshot), şi elimina problemele de pierdere a datelor care se scriu în baza de date de la momentul creării ultimei copii pana la distrugerea bazei de date. Pe lângă aceste riscuri, back-up-ul prin copiere integrala de tip snapshot solicita şi foarte mult spaţiu pe disk, la modul în care cu cat mai dese sunt copiile cu atât mai mult spaţiu se consuma.

HyperFiuse stochează datele integral pe fiecare nod de tip FULL NODE(vezi capitolul 3.3.7), iar de fiecare data când un nod de date de tip FULL NODE este adăugat într-o sub-reţea, acesta începe să se sincronizeze şi sa aducă toate datele din baza de date şi din blockchain. Procesul durează o perioada de timp direct proporţională cu volumul de date stocat. Nodul creat expune prin Interfaţa API toate funcţionalităţile prin care produsul software poate interacţiona cu HyperFiuse pentru a fi complet operaţional. Un nod pierdut sau deteriorat la care se conectează un produs software poate fi uşor reconstruit iar intre timp produsul software se poate conecta la oricare alt nod de tip FULL NODE din reţea.

Acest mecanism nu solicită nici un fel copiere integrala de tip snapshot, din contra, este nevoie de 2-3 noduri de tip FULL NODE care să asigure siguranţa maximă de care e nevoie pentru a garanta ca un sistem poate fi operaţional şi disponibil 100%, în comparaţie cu un back-up clasic care face cel puţin 2-3 copii integrale în fiecare zi.

3.3.17 Mecanismul de alerte şi notificări

Operaţiunile înregistrate în HyperFiuse, pot fi monitorizare de administratorul de sistem, iar la fiecare interacţiune de scriere, citire, ştergere sau actualizare a unei înregistrări, se pot defini alerte şi notificări. Mecanismul de alerte şi notificări se defineşte prin "momentul producerii unui eveniment - (CÂND)" şi "activitatea care trebuie sa se desfăşoare la producerea unui eveniment - (CE)".

Evenimentele care pot fi monitorizate în HyperFiuse sunt detaliate în tabelul următor (CÂND):

Operaţiune Scriere	**Eveniment**	**Descriere**
INSERT	BEFORE	Înainte de inserarea unei înregistrări se poate genera o alerta
INSERT	AFTER	După inserarea unui eveniment in sistem se va genera o alerta.
UPDATE	BEFORE	Înainte de actualizarea unei înregistrări se poate genera o alerta
UPDATE	AFTER	După actualizarea unui eveniment in sistem se va genera o alerta.
DELETE	BEFORE	Înainte de ştergerea unei înregistrări se poate genera o alerta
DELETE	AFTER	După ştergerea unui eveniment in sistem se va genera o alerta.
UPDATE-ID	BEFORE	Înainte de actualizarea unei anumite înregistrări se poate genera o alerta, alerta care nu se va genera decât dacă înregistrarea cu identificatorul unic specificat de administrator va fi actualizata
UPDATE-ID	AFTER	După actualizarea unui anumit eveniment in sistem se va genera o alerta, alerta care nu se va genera decât dacă înregistrarea cu identificatorul unic specificat de administrator va fi actualizata
DELETE-ID	BEFORE	Înainte de ştergerea unei anumite înregistrări se poate genera o alerta, alerta care nu se va genera decât dacă înregistrarea cu identificatorul unic specificat de administrator va fi ştearsă

DELETE-ID	AFTER	După ştergerea unui eveniment in sistem se va genera o alerta, alerta care nu se va genera decât dacă înregistrarea cu identificatorul unic specificat de administrator va fi ştearsă

Fiecare alerta generata de sistem in urma evenimentelor definite în tabelul de mai sus, poate fi operata în mai multe feluri după cum urmează (CE):

Tip prelucrare	**Parametru**	**Descriere**
Procedura Stocata	Înregistrare operata	La fiecare eveniment se executa un smart contract (vezi capitolul 3.3.19) de tip procedura stocata. Se va trimite ca parametru înregistrarea alterata şi procedura stocata va acţiona conform instrucţiunilor din smart contract asa cum au fost create de programator.
Mesaj Web	Înregistrare operata	La fiecare eveniment se va executa un request(cerere) web de tip POST, care are ca parametru înregistrarea alterata şi detalii despre tipul evenimentului care a înregistrat alerta. Va fi la latitudinea programului care va intercepta request-ul sa determine ce va face mai departe cu informaţiile primite.
Notificare Sistem	Înregistrare operata	La fiecare eveniment se va adăuga o înregistrare in tabela de notificări de sistem, înregistrare care va conţine detalii despre tipul evenimentului care a generat alerta, înregistrarea alterata şi utilizatorul care a iniţiat operaţiunea de scriere. Administratorul de sistem va observa aceste alerte in aplicaţia de administrare a sistemului.
E-mail	Înregistrare operata	La fiecare eveniment se va trimite un email cu detalii despre tipul evenimentului care a generat alerta, înregistrarea alterata şi utilizatorul care a iniţiat operaţiunea de scriere. E-mail-ul se va trimite numai dacă sistemul este configurat cu detaliile SMTP necesare trimiterii de email-uri din sistem.

3.3.18 Licenţierea nodurilor

HyperFiuse este un sistem creat pentru a deservi companiile care dezvolta soluţii software pe tehnologia blockchain, companii care îşi vor construi soluţiile pe o reţea privata. Mai exact, nodurile care fac partea din reţeaua HyperFiuse vor aparţine unui client, beneficiar, care deşi va administra întreg sistemul, poate apărea posibilitatea ca anumite noduri sa ruleze

la terţi. Din acest motiv, dar şi pentru a evita introducerea unui nod în reţea de către persoane neautorizate, este nevoie de un mecanism de licenţiere al nodurilor, un mecanism care sa permită crearea de noi noduri, dar în acelaşi timp acest proces sa fie supravegheat, aprobat şi controlat de beneficiarul final al sistemului.

Licenţierea nodurilor protejează beneficiarul de adăugarea unor noduri neaprobate în sistem, care pot fi folosite la extragerea de informaţii de către terţi neautorizaţi. Dacă în soluţiile blockchain tip public, nu este necesara aceasta protecţie, in HyperFiuse se doreşte ca nici un nod care este parte din reţea sa nu fie pornit fără aprobarea finala a beneficiarului de sistem.

Există două proceduri care trebuiesc urmate pentru a porni o reţea HyperFiuse: procedura de a porni nodul iniţial sau primul nod şi procedura de adăugare a altor noduri într-o reţea deja creata. Procedura de pornire sau de creare a nodului iniţial are următorii paşi:

1. Se porneşte nodul iniţial şi se da ca parametru de pornire numele/identificatorul unic al nodului.
2. Se apelează prin Interfaţa API funcţia */license/generateLicenseKeys* cu parametri *validity* si *startDate*, pentru a crea o licenţa care se prezintă sub forma unui cuplu de chei publice/private. Parametrul *validity* se refera la perioada de valabilitate a licenţei iar parametru *startDate* se refera la data în care licenţa intra în vigoare. API-ul generează şi stochează licenţa în sistem şi întoarce înregistrarea salvata cu identificatorul unic asociat acesteia.
3. Se apelează prin Interfaţa API funcţia */license/createNode* cu parametri *nodeId, licenseId* si *nodeType*, pentru a crea o configurare care asociază nodul iniţial cu licenţa creata la pasul anterior. Parametrul *nodeId* se încarcă cu numele nodului/identificatorul unic al acestuia, parametrul *licenseId* se încarcă cu identificatorul unic al licenţei care s-a generat la pasul 2 iar parametru *nodeType* specifica tipul nodului care este creat (nod de date sau nod de autorizare - pentru mai multe detalii vezi capitolul 3.2). API-ul asociază licenţa creata cu nodul pornit şi întoarce înregistrarea stocata pentru confirmare.
4. Se descarcă licenţa generată la pasul 2 prin Interfaţa API funcţia */license/download/{nodeName}* Parametru *nodeName* conţine numele nodului creat/identificatorul unic al acestuia.
5. Se copiază fişierul de licenţa descarcat pe nodul iniţial in folderul specific stocării de fişiere de licenţa

6. Se apelează prin Interfaţa API funcţia */license/applyConfiguration* care generează token-ul nodului care este folosit mai departe de nodul iniţial în comunicarea cu celelalte noduri din reţea (sub forma de cheie API).

Procedura de adăugare a unui nod nou în reţea este următoarea:

1. Se apelează prin Interfaţa API funcţia */license/generateLicenseKeys* cu parametri *validity* si *startDate*, pentru a crea o licenţa care se prezintă sub forma unui cuplu de chei publice/private. Parametrul *validity* se refera la perioada de valabilitate a licenţei iar parametru *startDate* se refera la data în care licenţa intra în vigoare. API-ul generează şi stochează licenţa în sistem şi întoarce înregistrarea salvata cu identificatorul unic asociat acesteia.
2. Se apelează prin Interfaţa API funcţia */license/createNode* cu parametri *nodeId, licenseId* si *nodeType*, pentru a crea o configurare care asociază nodul nou cu licenţa creata la pasul anterior. Parametrul *nodeId* se încarcă cu numele nodului/identificatorul unic al acestuia, parametrul *licenseId* se încarcă cu identificatorul unic al licenţei care s-a generat la pasul 1 iar parametru *nodeType* specifica tipul nodului care este creat (nod de date sau nod de autorizare - pentru mai multe detalii vezi capitolul 3.2). API-ul asociază licenţa creata cu nodul nou şi întoarce înregistrarea stocata pentru confirmare.
3. Se descarca licenţa generata la pasul 1 prin Interfaţa API funcţia */license/download/{nodeName}* Parametru *nodeName* conţine numele nodului nou creat/identificatorul unic al acestuia.
4. Se copiază fişierul de licenţa descarcat pe nodul nou creat în folderul specific stocării de fişiere de licenţa
5. Se porneşte nodul nou creat cu parametrul de pornire numele nodului/identificatorul unic al acestuia ales de operator la pasul 2. La pornirea nodului, se generează automat un token specific noului nod creat, care va folosi în comunicarea cu alte noduri din reţeaua HyperFiuse (cheie secreta API)

Mecanismele definite au ca scop clarificarea faptului ca exista un mecanism de licenţiere în baza căruia un nod poate fi asociat cu o reţea HyperFiuse, clarificarea procedurii prin care acest mecanism de licenţiere este activat împreună cu pornirea şi adăugarea unui nod nou, precum şi clarificarea faptului ca comunicarea intre nodurile reţelei se face prin protocolul TCP şi numai în baza unei chei secrete API care este asociata cu licenţa de funcţionare a fiecărui nod în parte.

3.3.19 Contracte Inteligente (Smart Contracts)

Contractele inteligente sau smart contracts, sunt secvente de cod (programe software) pe care un programator le poate alipi sau adăuga sistemului HyperFiuse. Aceste programe au ca scop extinderea funcţionalităţilor existente ale platformei, permiţând programatorului sau beneficiarului sa adauge funcţii noi pe lângă caracteristicile standard care sunt disponibile la instalare. De asemenea, contractele inteligente se mai evidenţiază şi prin faptul ca sunt stocate în blockchain, iar codul sursa care le defineşte devine imutabil, oferind beneficiarului sau terţilor care interacţionează cu sistemul HyperFiuse, garanţia ca anumite funcţionalităţi nu pot fi modificate nici măcar de beneficiarul platformei.

Dezvoltarea de contracte inteligente este la îndemâna oricărui programator, care are la dispoziţie o instanta de HyperFiuse, codul sursa al contractului inteligent fiind în proprietatea programatorului, acesta fiind liber de a comercializa sau pune la dispoziţie gratuit funcţionalitatea creata. Cu toate acestea, mecanismul şi mediul de execuţie al contractelor inteligente, adică platforma HyperFiuse este şi va rămâne proprietatea intelectuala a Ingenium Blockchain Technologies, creatorul HyperFiuse. De asemenea, pentru a crea un contract inteligent, programatorul trebuie sa urmeze procedurile de dezvoltare puse la dispoziţie de Ingenium Blockchain Technologies, precum şi librăriile ajutătoare şi/sau platformele care facilitează aceste dezvoltări.

HyperFiuse este un mediu de execuţie al contractelor inteligente, iar prin intermediul unui contract inteligent, programatorul poate interacţiona cu mai multe componente/funcţionalităţi deja existente precum Criptarea/Decriptarea, verificarea integrităţii unei înregistrări şamd. La momentul dezvoltării unui contract inteligent, programatorul are nevoie de interfeţele de comunicare cu funcţionalităţile existente, chiar dacă nu se afla într-un mediu de execuţie HyperFiuse. Practic, prin intermediul librăriilor puse la dispoziţie de Ingenium Blockchain Technologies, un programator poate folosi un mediu de dezvoltare Java pentru crea contracte inteligente, fără a avea neapărat la dispoziţie un sistem HyperFiuse funcţional. Într-adevăr, pentru testarea contractului dezvoltat, va fi nevoie de un mediu de test sau o instanta HyperFiuse pentru rularea funcţionalităţii create. Aceste librarii cu interfeţe de interacţionare HyperFiuse sunt disponibile la descărcare pentru orice programator care doreşte sa dezvolte contracte inteligente.

Contractele inteligente sunt dezvoltate pe calculatorul personal de către programatori, iar odată terminate acestea pot fi încărcate şi activate în sistemul HyperFiuse. Codul sursă al unui contract inteligent va fi în format arhiva Java (*.jar), şi poate conţine sursele programului,

ataşarea acestora la arhiva fiind strict la latitudinea programatorului. Încărcarea contractului inteligent în HyperFiuse se face prin intermediul interfeţei de administrare de către administratorul sistemului sau de către programator, în funcţie de drepturile de acces ale acestuia. Odată încărcat un contract inteligent, HyperFiuse va stoca conţinutul arhivei precum şi componentele acesteia şi va genera semnătura calculata printr-o funcţie hash a oricărui fişier care compune arhiva încărcata. Toate semnăturile asociate fişierelor care compun arhiva contractului inteligent, împreuna cu operaţia de încărcare în sine, sunt stocate pe blockchain (lanţul imutabil) pentru a putea verifica la orice moment din timp integritatea funcţionalităţilor pe care contractul inteligent le expune, şi pentru a garanta utilizatorilor sistemului ca codul de executat nu a fost alterat.

Mecanismul de distribuţie date (vezi capitolul 3.3.2) va ajuta ca contractul inteligent încărcat în sistem sa fie distribuit şi încărcat pe toate nodurile de date din reţea, în asa fel încât funcţionalitatea pe care contractul inteligent o expune sa fie disponibila în tot sistemul. Acest procedeu ajuta foarte mult şi la mecanismele de propagare de cod şi funcţionalităţi noi într-un sistem aflat în producţie.

Un contract inteligent se supune şi mecanismului de versionare al HyperFiuse, mecanism care permite unui programator sa aducă îmbunătăţiri unui contract inteligent, încărcând versiuni noi ale contractului şi permiţând rularea şi execuţia unui contract inteligent cu parametru de versiune. Programatorul sau administratorul sistemului poate opta şi pentru anularea unei versiuni de contract inteligent, pentru situaţiile în care versiunile noi sunt critice sau nu se mai doreşte expunerea unor anumite funcţionalităţi.

Un contract inteligent se încarcă în sistemul HyperFiuse şi la publicare primeşte un id unic ca răspuns la unei operaţiuni de încărcare efectuate corect. Încărcarea unui contract inteligent poate eşua în contextul unor erori de cod sursa sau în contextul în care se rulează anumite teste de securitate fără rezultat pozitiv. Lista contractelor publicate de un utilizator, în general un administrator are drept de încărcare a unui smart contract, este disponibila în aplicaţia de administrare a sistemului. Fiecare contract inteligent disponibil în lista contractelor încărcate poate fi operat de administrator în felul următor:

- activare/dezactivare - contractul inteligent odată încărcat nu este activ, disponibil funcţional pana când nu este activat de administrator. Dezactivarea unui contract inteligent duce contractul în starea în care nu mai poate fi accesat de utilizatori, contractul devenind nefuncţional
- ştergere - contractul inteligent este anulat, şi nu mai poate fi apelat de nici un utilizator. Operaţiunea este ireversibila

- actualizare - se încarcă o noua versiune a contractului inteligent
- verificare istoric - se verifica istoricul apelurilor utilizatorilor la contractul inteligent. Istoricul se prezintă sub forma unui log imutabil care detaliază apelurile utilizatorilor şi cuprinde informaţii precum metoda din contract apelata, parametri de apelare, utilizatorul care a apelat contractul, data şi ora evenimentului, eventuale erori precum şi log-urile specificate de programator în interiorul contractului.
- activare/dezactivare sistem de logare contract inteligent
- definire contract inteligent ca activitate recurenta cu parametri globali (data, ora, etc) - se aplica numai pentru contracte inteligente de tip procedura stocata
- specificare utilizator în numele căruia se executa contractul inteligent - se aplica numai pentru contracte inteligente de tip procedura stocata

Contractele inteligente sunt de mai multe tipuri şi se diferenţiază prin natura în care se executa. Avem contracte inteligente de tip API şi contracte inteligente de tip procedura stocata.

Contractele inteligente de tip API, sunt contractele a căror funcţionalitate este accesata direct de programator prin Interfaţa API. Un contract inteligent este apelat prin Interfaţa API, prin intermediul unui API general, care ia ca parametri in URL identificatorul unic al contractului, versiunea contractului şi metoda din contract care se apelează. Cererea web API este de tip POST şi în conţinutul cererii se transmit parametri metodei apelate în format JSON. Răspunsul cererii web de apelare a unei metode unui contract inteligent poate fi de orice natura specifica unei cereri de tip POST: JSON, text, HTML, descărcare de fişier şamd.

Contractele inteligente de tip procedura stocata, sunt contractele care nu pot fi apelate direct de programator prin intermediul Interfeţei API, dar pot fi apelate de mecanismul de alertare sau de alte contracte inteligente. Scopul acestora este sa faciliteze funcţionalitate în interiorul HyperFiuse, care poate fi reutilizata şi de alte contracte sau poate fi apelata de procese interne HyperFiuse (asa cum este mecanismul de alerte care în urma unui eveniment poate apela un contract inteligent). Apelarea unei metode dintr-un contract inteligent de tip procedura stocata, întoarce rezultatul la momentul execuţiei, iar log-urile asociate pot fi vizualizate de administrator în interfaţa de administrare. Contractele inteligente de tip procedura stocata pot fi definite de către administrator şi ca activităţi recurente, procese care se execută regulat şi efectuează anumite operaţiuni la intervale regulate de timp.

Un contract inteligent, odată încărcat în sistemul HyperFiuse, poate fi instruit de programatorul care l-a creat sa efectueze cel puţin următoarele operaţiuni:

- poate opera asupra oricărei entităţi/tabele de date. Operarea unei entităţi se refera la modificarea structurii acesteia sau la manipularea datelor (citire, adăugare, actualizare şi ştergere)
- poate efectua operaţiuni de verificare a integrităţii datelor
- poate efectua operaţiuni de recuperare a datelor
- poate efectua operaţiuni de criptare/decriptare de date
- poate efectua operaţiuni de semnare digitala a datelor
- poate accesa diferite versiuni ale datelor
- poate solicita acces la date şi poate oferi acces la date
- poate apela contracte inteligente de tip procedura stocata
- poate accesa servicii expuse de HyperFiuse: trimitere email, generare de alerte notificări, scriere de loguri etc
- poate accesa librăriile de funcţionalităţi puse la dispoziţie de HyperFiuse: librarii pentru validarea parametrilor de intrare, librarii pentru calcule matematice, librarii pentru formatare text
- poate accesa servicii web externe
- poate accesa orice funcţionalitate pusa la dispoziţie de limbajul de programare Java, în nivelului de securitate aplicat de administratorul platformei

Toate metodele dintr-un contract inteligent se vor executa în numele unui utilizator de sistem, şi toate restricţiile de securitate respectiv drepturile de acces pe care acesta le are, se vor aplica pe parcursul execuţiei metodei din contract. Utilizatorul care apelează contractul inteligent este utilizatorul ale cărui drepturi de acces vor fi aplicate. Pentru anumite proceduri stocate recurente, poate fi configurata execuţia în numele unui utilizator anume sau utilizator de sistem (administrator)

3.3.20 GraphQL

GraphQL este un limbaj open-source de interogare a datelor obţinute prin interfeţe de tip API şi un mediu de manipulare a acestora. Limbajul a fost dezvoltat în 2012 de Facebook pentru uz intern, şi a fost făcut public în 2015. GraphQL se prezintă tot ca o interfaţă API care permite operaţiuni de manipulare, mult mai complexe a datelor, cu API-uri care sunt limitate prin operaţiunile pe care le pun la dispoziţia programatorilor.

HyperFiuse expune prin Interfaţa API, operaţiunile primare de interacţiune cu datele salvate în sistem: creare date, actualizare date, ştergere date şi citire de date. Cu ajutorul

GraphQL, Interfaţa API HyperFiuse oferă posibilitatea programatorului sa creeze interogări complexe cu aceste date într-un mod mai uşor şi mai eficient.

GraphQL nu este proprietatea HyperFiuse, este doar o componenta open-source pe care acesta o foloseşte, singurul lucru care trebuie reţinut din acest context este ca HyperFiuse, cu ajutorul GraphQL, oferă posibilitatea programatorului de a rula şi interogări complexe prin Interfaţa REST API.

Invenţia a fost în principal descrisă mai sus cu referire la câteva exemple de realizare. Cu toate acestea, aşa cum este uşor de apreciat de o persoană de specialitate în domeniu, alte exemple de realizare faţă de cele dezvăluite mai sus sunt la fel de posibile în interiorul scopului invenţiei, aşa cum este definit prin revendicările anexate.

REVENDICĂRI

1. Platformă de conectare destinată să conecteze un motor de blockchain cu o bază de date tradiţională, platforma fiind implementată sub forma unei reţele de noduri, reţeaua de noduri menţionată fiind împărţită în cel puţin două subreţele: o subreţea de securitate şi o subreţea de date, toate nodurile din sub-reţeaua de securitate conţinând informaţii referitoare la cheile de securitate ale utilizatorilor, licenţele de funcţionare, politice de acces, precum şi alte informaţii care ţin de mecanismul de licenţiere, autentificare şi autorizare al utilizatorilor care accesează platforma, şi în care nodurile din cel puţin o subreţea de date cuprind o componentă software care foloseşte o reţea de calculatoare, o interfaţă de comunicare API ce permite interacţiunea cu reţeaua de calculatoare şi preluarea informaţiilor care trebuie salvate în sistemul de stocare, o interfaţă GraphQL de interogare a datelor, un motor procesare date, un motor blockchain, o interfaţa de conectare între motorul de procesare date şi motorul blockchain, şi o bază de date.

2. Platformă conform revendicării 1, cuprinzând mijloace de validare a datelor şi salvarea acestora în baza de date menţionată, precum şi salvarea în blockchain a unei referinţe despre locaţia datelor în baza de date şi a unei amprente digitale a acestor informaţii.

3. Platformă conform revendicării 1, cuprinzând mijloace de calcul a amprentei digitale a înregistrării salvate folosind o funcţie hash (Ex: MD5, SHA256, SHA512, dar pot fi folosite şi alte funcţii de calcul a amprentei digitale).

4. Platformă conform revendicării 1, cuprinzând mijloace de scriere la nivel de bază de date, în care mijloacele de scriere pot fi de trei feluri: inserare, modificare şi ştergere, fiecare operaţiune fiind stocată în blockchain ca tranzacţie independentă alături de informaţiile de referinţă, precum o tabelă, un identificator al înregistrării şi amprentă digitală menţionată.

5. Platformă conform revendicării 1, în care motorul de blockchain are autoritatea de distribuţie a datelor din blockchain pe toate nodurile din toate sub reţelele de date, iar datele din baza de date sunt sincronizate cu ajutorul platformei de conectare, fiecare nod din sub reţeaua de date este actualizat în blockchain cu blocurile cele mai recent validate de motorul de blockchain, iar odată sincronizat un nou bloc, platforma de conectare procesează fiecare tranzacţie din bloc şi pentru fiecare dintre acestea iniţiază o cerere de obţinere a datelor modificate către nodul emitent în care a fost creată tranzacţia, iar odată primite datele, acestea

sunt modificate şi în baza de date a nodului recipient conform tipului de tranzacţie specificat în blockchain.

6. Platformă conform revendicării 1, în care platforma de conectare permite accesul direct la baza de date, respectiv publicarea detaliilor de conectare directă la motorul de baze de date.

7. Platformă conform revendicării 1, în care platforma de conectare permite accesul la baza de date prin interfaţa API, interfaţă care va returna datele în format decriptat, va filtra datele în funcţie de drepturile pe care utilizatorul care efectuează citirea le are asupra acestora, va efectua verificări de integritate asupra datelor şi va permite citirea cu opţiunea de reconstruire a datelor în cazul în care au fost identificate modificări/ştergeri neautorizate asupra acestora.

8. Platformă conform revendicării 1, în care sub reţelele de date pot fi configurate pentru a limita sincronizarea datelor din anumite tabele/colecţii la nodurile din cadrul subretelei, poliţele de sincronizare a datelor aplicându-se la nivel de subreţea, unde pentru fiecare subreţea de date se poate specifica ce tabelă/colecţie de date să nu se sincronizeze cu noduri din afara sub reţelei.

9. Platformă conform revendicării 1, în care platforma permite configurarea unui nod în aşa fel încât acesta să nu sincronizeze toate datele, şi să stocheze numai datele care au fost salvate folosind Interfaţa API instalată pe nodul menţionat.

10. Platformă conform revendicării 1, în care platforma permite criptarea informaţiilor printr-un cuplu de chei format dintr-o cheie publică şi o cheie privată, cheia privată a unui utilizator fiind protejată prin 3 mecanisme diferite şi anume: dispozitiv hardware extern care stochează cheia privată şi este în posesia utilizatorului, păstrarea cheii private de către utilizator în format şir de caractere digital sau în care cheile sunt stocate în sub reţeaua de securitate şi gestionate de platformă, şi a treia varianta în care cheile private sunt gestionate de platformă, sunt destinate produselor software care nu au utilizatori pregătiţi să îşi protejeze o cheie privată şi pun accent pe alte funcţionalităţi oferite de platformă.

11. Platformă conform revendicării 1, în care platforma cuprinde mijloace de licenţiere a nodurilor care permite crearea de noi noduri, sub supravegherea, aprobarea şi controlul utilizatorului de platformă.

12. Metodă de conectare a unui motor de blockchain cu cel puţin un motor de management a bazelor de date, metoda cuprinzând etapele de asigurare a unei platforme de conectare, conform revendicării 1, platforma menţionată fiind implementată sub forma unei reţele de noduri împărţită în cel puţin două subreţele: o subreţea de securitate şi o subreţea de date, toate nodurile din sub reţeaua de securitate conţinând informaţii referitoare la cheile de securitate ale utilizatorilor, licenţele de funcţionare, politice de acces, precum şi alte informaţii care ţin de mecanismul de licenţiere, autentificare şi autorizare al utilizatorilor care accesează platforma, şi în care nodurile din acea cel puţin o subreţea de date cuprind o componentă software care foloseşte o reţea de calculatoare, o interfaţă de comunicare API ce permite interacţiunea cu reţeaua de calculatoare şi preluarea informaţiilor care trebuiesc salvate în sistemul de stocare, o interfaţă GraphQL de interogare a datelor, un motor procesare date, un motor blockchain, o interfaţa de conectare între motorul de procesare date şi motorul blockchain, şi o bază de date.

13. Metodă conform revendicării 12, cuprinzând etapele de validare a datelor şi salvare a acestora în baza de date menţionată, precum şi salvarea în blockchain a unei referinţe despre locaţia datelor în baza de date şi a unei amprente digitale a acestor informaţii.

14. Metodă conform revendicării 12, cuprinzând o etapă de calcul a amprentei digitale a înregistrării salvate folosind o funcţie hash (Ex: MD5, SHA256, SHA512, dar pot fi folosite şi alte funcţii de calcul a amprentei digitale).

15. Metodă conform revendicării 12, cuprinzând etapa de scriere la nivel de bază de date, în care etapa de scriere pot fi de trei feluri: inserare, modificare şi ştergere, fiecare operaţiune fiind stocată în blockchain ca şi tranzacţie independentă alături de informaţiile de referinţă, precum o tabelă, un identificator al înregistrării şi amprentă digitală menţionată.

16. Metodă conform revendicării 12, în care autoritatea de distribuţie a datelor din blockchain pe toate nodurile din toate sub-reţelele de date aparţine motorului de blockchain, iar datele din baza de date sunt sincronizate cu ajutorul platformei de conectare, fiecare nod din sub-reţeaua de date este actualizat în blockchain cu blocurile cele mai recent validate de motorul de blockchain, iar odată sincronizat un nou bloc, platforma de conectare procesează fiecare tranzacţie din bloc şi pentru fiecare dintre acestea iniţiază o cerere de obţinere a datelor

modificate către nodul emitent în care a fost creată tranzacţia, iar odată primite datele, acestea sunt modificate şi în baza de date a nodului recipient conform tipului de tranzacţie specificat în blockchain.

17. Metodă conform revendicării 12, cuprinzând etapa de permitere a accesului direct la baza de date, respectiv publicarea detaliilor de conectare directă la motorul de baze de date.

18. Metodă conform revendicării 12, cuprinzând etapa de permitere a accesului la baza de date prin interfaţa API, interfaţă care va returna datele în format decriptat, va filtra datele în funcţie de drepturile pe care utilizatorul care efectuează citirea le are asupra acestora, va efectua verificări de integritate asupra datelor şi va permite citirea cu opţiunea de reconstruire a datelor în cazul în care au fost identificate modificări/ştergeri neautorizate asupra acestora.

19. Metodă conform revendicării 12, cuprinzând etapa de configurare a sub-reţelelor de date pentru a limita sincronizarea datelor din anumite tabele/colecţii la nodurile din cadrul sub-reţelei, poliţele de sincronizare a datelor aplicându-se la nivel de subreţea, unde pentru fiecare subreţea de date se poate specifica ce tabelă/colecţie de date să nu se sincronizeze cu noduri din afara sub-reţelei.

20. Metodă conform revendicării 12, cuprinzând etapa de configurare a unui nod în aşa fel încât acesta să nu sincronizeze toate datele, şi să stocheze numai datele care au fost salvate folosind Interfaţa API instalată pe nodul menţionat.

21. Metodă conform revendicării 12, cuprinzând etapa de criptare a informaţiilor printr-un cuplu de chei format dintr-o cheie publică şi o cheie privată, cheia privată a unui utilizator fiind protejată prin 3 mecanisme diferite şi anume: dispozitiv hardware extern care stochează cheia privată şi este în posesia utilizatorului, păstrarea cheii private de către utilizator în format şir de caractere digital sau în care cheile sunt stocate în sub-reţeaua de securitate şi gestionate de platformă, şi a treia varianta în care cheile private sunt gestionate de platformă, sunt destinate produselor software care nu au utilizatori pregătiţi să îşi protejeze o cheie privată şi pun accent pe alte funcţionalităţi oferite de platformă.

22. Metodă conform revendicării 12, cuprinzând etapa de licenţiere a nodurilor care permite crearea de noi noduri, sub supravegherea, aprobarea şi controlul utilizatorului de platformă.

23. Metodă conform revendicării 22, în care etapa de pornire sau de creare a unui nod iniţial cuprinde următorii paşi:

- pornirea nodul iniţial şi stabilirea ca parametru de pornire numele/identificatorul unic al nodului;
- apelarea prin Interfaţa API a funcţiei */license/generateLicenseKeys* cu parametri *validity* si *startDate*, pentru a crea o licenţă care se prezintă sub forma unui cuplu de chei publice/private
- apelarea prin interfaţa API a funcţiei */license/createNode* cu parametri *nodeId, licenseId* si *nodeType*, pentru a crea o configuraţie care asociază nodul iniţial cu licenţa creată la pasul anterior;
- descărcarea licenţei generată la pasul 2, prin interfaţa API, funcţiei */license/download/{nodeName}*
- copierea fişierul de licenţă descărcat pe nodul iniţial în folderul specific stocării de fişiere de licenţă;
- apelarea prin Interfaţa API a funcţiei */license/applyConfiguration* care generează token-ul nodului care este folosit mai departe de nodul iniţial în comunicarea cu celelalte noduri din reţea.

24. Metodă conform revendicării 22, în care etapa de adăugare a unui nod nou în reţea cuprinde următorii paşi:

- apelarea prin interfaţa API a funcţiei */license/generateLicenseKeys* cu parametri *validity* şi *startDate*, pentru a crea o licenţă care se prezintă sub forma unui cuplu de chei publice/private;
- apelarea prin interfaţa API a funcţiei */license/createNode* cu parametri *nodeId, licenseId* si *nodeType*, pentru a crea o configuraţie care asociază nodul nou cu licenţa creată la pasul anterior;
- descărcarea licenţei, generată la pasul 1 prin Interfaţa API, a funcţiei */license/download/{nodeName};*
- copierea fişierului de licenţă descărcat pe nodul nou creat în folderul specific stocării de fişiere de licenţă;
- pornirea nodului nou creat cu parametrul de pornire numele nodului/identificatorul unic al acestuia ales de operator la pasul 2.

25. Produs program de calculator pentru conectarea unui motor de blockchain cu cel puţin un motor de management a bazelor de date, programul de calculator cuprinzând un cod de

program de calculator care, atunci când este rulat pe un echipament de utilizator, determină echipamentul de utilizator să realizeze metoda din oricare dintre revendicările 12 la 24.

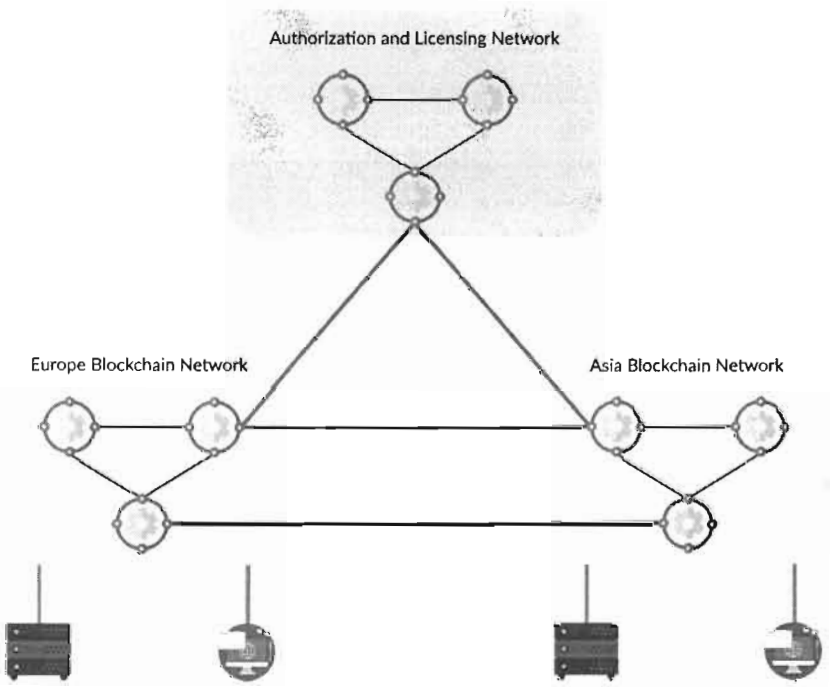
Authorization and Licensing Network
Europe Blockchain Network
Asia Blockchain Network

Figura 1

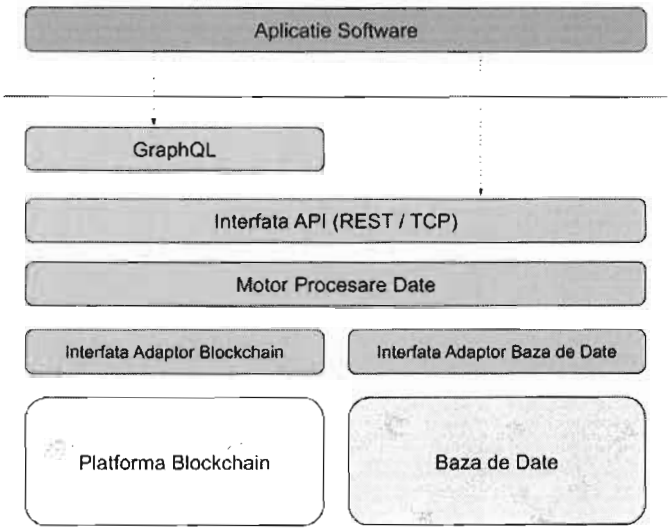
Aplicatie Software
GraphQL
Interfata API (REST / TCP)
Motor Procesare Date
Interfata Adaptor Blockchain
Interfata Adaptor Baza de Date
Platforma Blockchain
Baza de Date

Figura 2

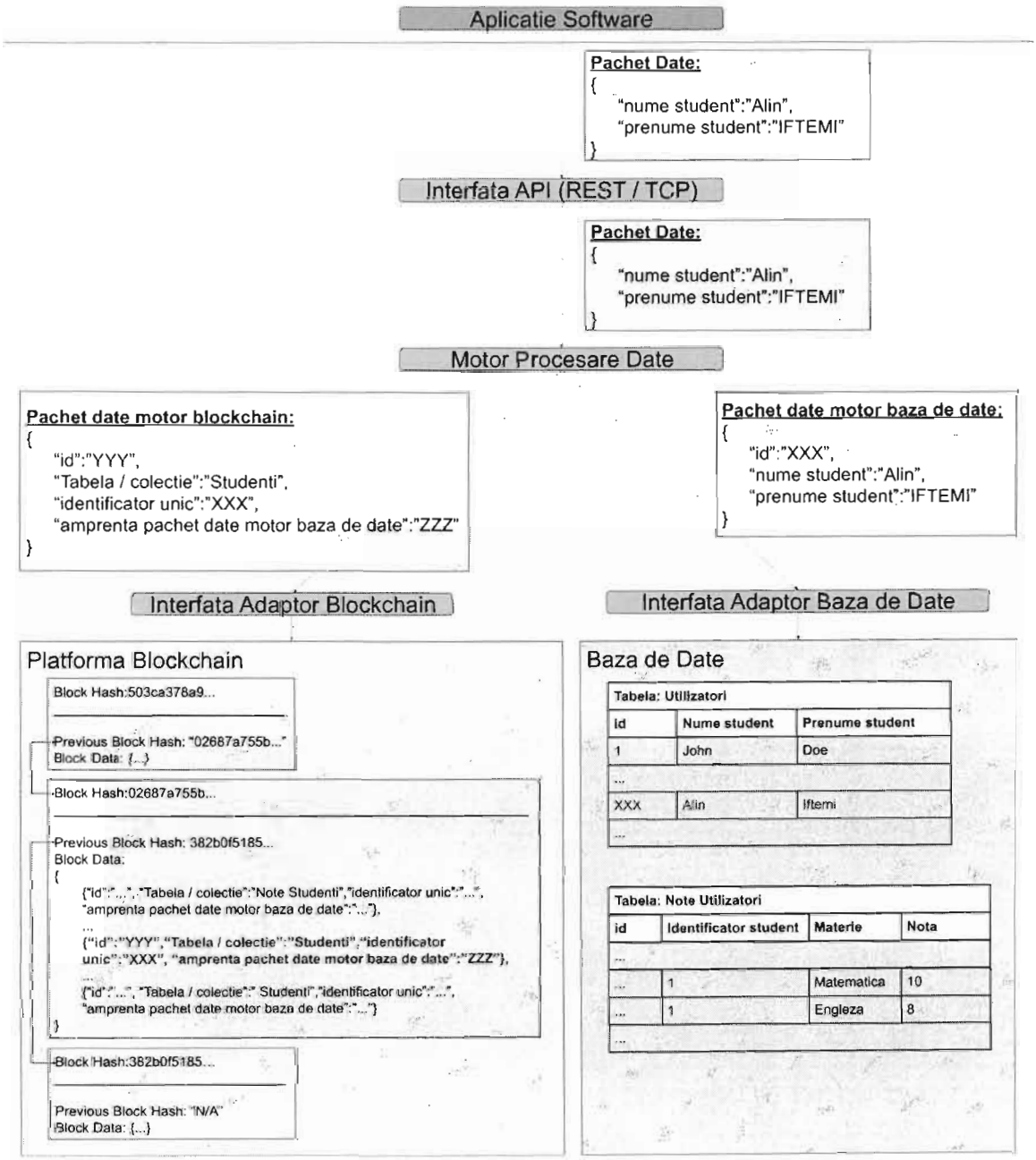
Aplicatie Software
Pachet Date:
{
"nume student":"Alin",
"prenume student":"IFTEMI"
}
Interfata API (REST / TCP)
Pachet Date:
{
"nume student":"Alin",
"prenume student":"IFTEMI"
}
Motor Procesare Date
Pachet date motor blockchain:
{
"id":"YYY",
"Tabela / colectie":"Studenti",
"identificator unic":"XXX",
"amprenta pachet date motor baza de date":"ZZZ"
}
Pachet date motor baza de date:
{
"id":"XXX",
"nume student":"Alin",
"prenume student":"IFTEMI"
}
Interfata Adaptor Blockchain
Interfata Adaptor Baza de Date
Platforma Blockchain
Block Hash:503ca378a9...
Previous Block Hash: "02687a755b..."
Block Data: {...}
Block Hash:02687a755b...
Previous Block Hash: 382b0f5185...
Block Data:
{
{"id":"...", "Tabela / colectie":"Note Studenti","identificator unic":"...",
"amprenta pachet date motor baza de date":"..."},
...
{"id":"YYY","Tabela / colectie":"Studenti","identificator
unic":"XXX", "amprenta pachet date motor baza de date":"ZZZ"},
...
{"id":"...", "Tabela / colectie":" Studenti","identificator unic":"...",
"amprenta pachet date motor baza de date":"..."}
}
Block Hash:382b0f5185...
Previous Block Hash: "N/A"
Block Data: {...}
Baza de Date
Tabela: Utilizatori
id
Nume student
Prenume student
1
John
Doe
XXX
Alin
Iftemi
Tabela: Note Utilizatori
id
Identificator student
Materie
Nota
1
Matematica
10
1
Engleza
8

Figura 3

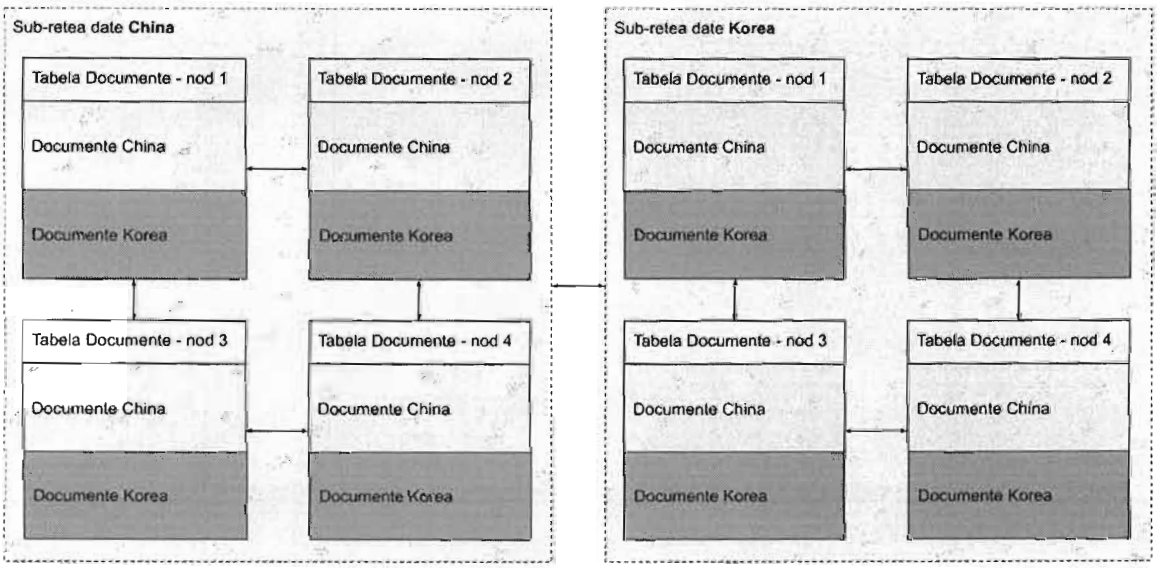
Sub-retea date China
Tabela Documente - nod 1
Documente China
Documente Korea
Tabela Documente - nod 2
Documente China
Documente Korea
Tabela Documente - nod 3
Documente China
Documente Korea
Tabela Documente - nod 4
Documente China
Documente Korea
Sub-retea date Korea
Tabela Documente - nod 1
Documente China
Documente Korea
Tabela Documente - nod 2
Documente China
Documente Korea
Tabela Documente - nod 3
Documente China
Documente Korea
Tabela Documente - nod 4
Documente China
Documente Korea

Figura 4

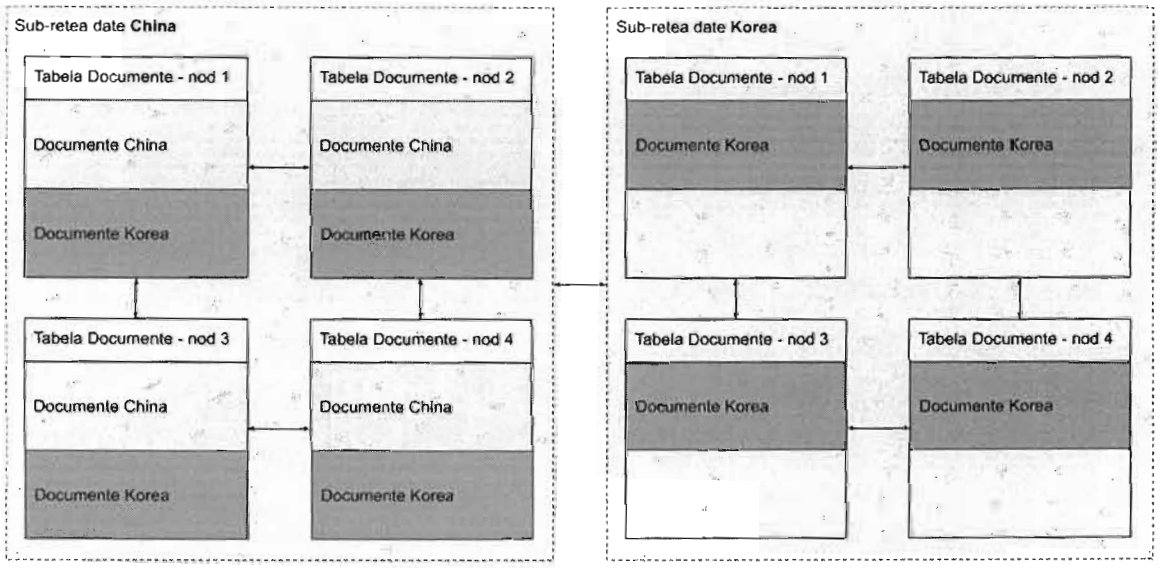
Sub-retea date China
Tabela Documente - nod 1
Documente China
Documente Korea
Tabela Documente - nod 2
Documente China
Documente Korea
Tabela Documente - nod 3
Documente China
Documente Korea
Tabela Documente - nod 4
Documente China
Documente Korea
Sub-retea date Korea
Tabela Documente - nod 1
Documente Korea
Tabela Documente - nod 2
Documente Korea
Tabela Documente - nod 3
Documente Korea
Tabela Documente - nod 4
Documente Korea

Figura 5

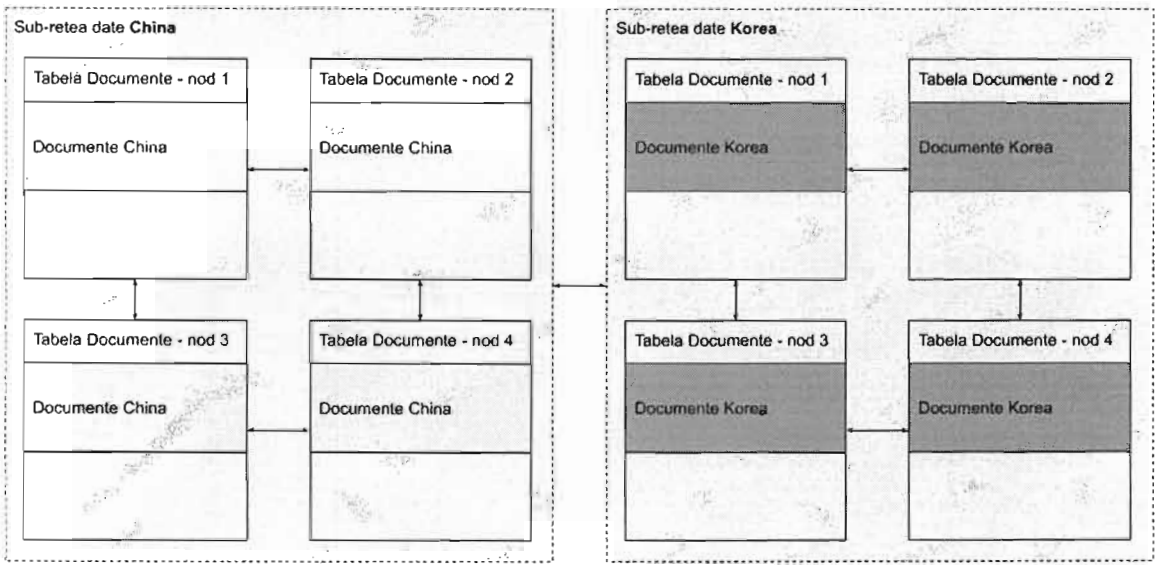
Sub-retea date China
Tabela Documente - nod 1
Documente China
Tabela Documente - nod 2
Documente China
Tabela Documente - nod 3
Documente China
Tabela Documente - nod 4
Documente China
Sub-retea date Korea
Tabela Documente - nod 1
Documente Korea
Tabela Documente - nod 2
Documente Korea
Tabela Documente - nod 3
Documente Korea
Tabela Documente - nod 4
Documente Korea

Figura 6

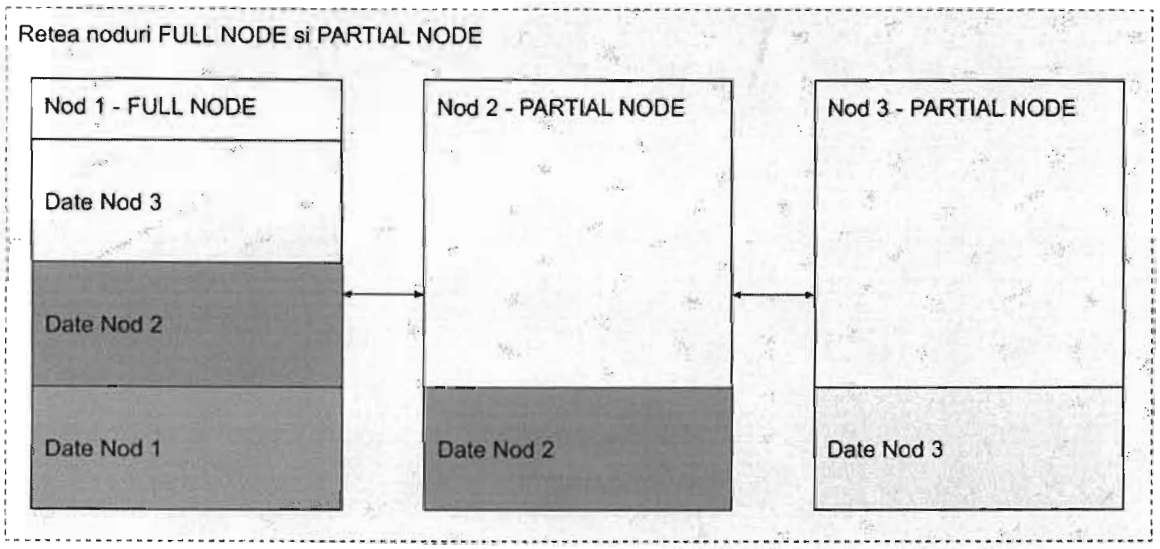
Retea noduri FULL NODE si PARTIAL NODE
Nod 1 - FULL NODE
Date Nod 3
Date Nod 2
Date Nod 1
Nod 2 - PARTIAL NODE
Date Nod 2
Nod 3 - PARTIAL NODE
Date Nod 3

Figura 7