**TRASABILITATEA PRODUSELOR ALIMENTARE FOLOSIND BLOCKCHAIN**

LUCRARE DE LICENŢĂ

|  |  |  |
| --- | --- | --- |
|  | Absolvent: | **Bogdan-Nicolae PUPEZĂ** |
|  |  |  |
|  | Coordonator științific: | **Prof. dr. ing.**  **Sebestyen-Pal GHEORGHE** |

**2024**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| DECAN, |  | | DIRECTOR DEPARTAMENT, | |
| **Prof. dr. ing. Liviu MICLEA** |  | **Prof. dr. ing. Rodica POTOLEA** | |

Absolvent: **Bogdan-Nicolae PUPEZĂ**

**TRASABILITATEA PRODUSELOR ALIMENTARE FOLOSIND BLOCKCHAIN**

1. **Enunțul temei:** *Scurtă descriere a temei lucrării de licență și datele inițiale*
2. **Conținutul lucrării:** *(enumerarea pârților componente) Exemplu: Pagina de prezentare, aprecierile coordonatorului de lucrare, titlul capitolului 1, titlul capitolului 2,… titlul capitolului n, bibliografie, anexe.*
3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Tehnologia Informației
4. **Consultanți**: Prof. dr. ing. Sebestyen-Pal Gheorghe
5. **Data emiterii temei:** 1 noiembrie 2023
6. **Data predării:** 8 iulie 2024

|  |  |
| --- | --- |
| Absolvent: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
| Coordonator științific: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Declarație pe propria răspundere privind**

**autenticitatea lucrării de licență**

Subsemnatul(a)**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**, legitimat(ă) cu \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ seria \_\_\_\_\_\_\_ nr. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
CNP \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, autorul lucrării \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_elaborată în vederea susținerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ din cadrul Universității Tehnice din Cluj-Napoca, sesiunea \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ a anului universitar \_\_\_\_\_\_\_\_\_\_, declar pe propria răspundere că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

In cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

|  |  |  |
| --- | --- | --- |
| Data  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Nume, Prenume  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |
|  |  | Semnătura |

Cuprins

[Capitolul 1. Introducere 1](#_Toc171166610)

[Capitolul 2. Obiectivele proiectului 3](#_Toc171166611)

[**2.1.1.** Obiectivul principal 3](#_Toc171166612)

[**2.1.2.** Obiective secundare 3](#_Toc171166613)

[**2.1.3.** Cerinte functionale 3](#_Toc171166614)

[**2.1.4.** Cerinte non-functionale 4](#_Toc171166615)

[Capitolul 3. Studiu bibliografic 5](#_Toc171166616)

[3.1. Sisteme de trasabilitate traditionale 5](#_Toc171166617)

[3.2. Sisteme de trasabilitate bazate pe Blockchain 5](#_Toc171166618)

[**3.2.1.** Integrarea tehnologiei blockchain in pasii procesului de trasabilitate 5](#_Toc171166619)

[**3.2.2.** Arhitecturi folosite in sistemele de trasabilitate 6](#_Toc171166620)

[**3.2.3.** Studii de caz si aplicatii blockchain practice in sistemele de trasabilitate 7](#_Toc171166621)

[**3.2.4.** Integrarea blockchain-ului cu alte tehnologii 9](#_Toc171166622)

[**3.2.5.** Provocari si vulnerabilitati intalnite 9](#_Toc171166623)

[**3.2.6.** Concluzia rezultata pe baza articolelor in ceea ce priveste utilizarea tehnologiei blockchain in sistemele de trasabilitate 9](#_Toc171166624)

[Capitolul 4. Analiză și fundamentare Teoretică 11](#_Toc171166625)

[4.1. Concepte de baza in trasabilitate 11](#_Toc171166626)

[4.2. Cazuri de utilizare ale sistemului de trasabilitate 11](#_Toc171166627)

[4.3. Arhitectura aplicatiei 12](#_Toc171166628)

[4.4. Salvarea elementelor de date cheie in contractele inteligente 14](#_Toc171166629)

[4.5. Etapa de transformare a materiilor prime in produse 15](#_Toc171166630)

[4.6. Inregistrarea unui transfer de proprietate 17](#_Toc171166631)

[4.7. Functia de reconstructie a traseului unui produs 18](#_Toc171166632)

[4.8. Verificarile din cadrul sistemului de trasabilitate 20](#_Toc171166633)

[4.9. Anularea unei actiuni 21](#_Toc171166634)

[4.10. Semnalarea unui produs contaminat in sistemul de trasabilitate 21](#_Toc171166635)

[Capitolul 5. Proiectare de detaliu și implementare 24](#_Toc171166636)

[5.1. Back-end: Contractele inteligente ale sistemului de trasabilitate 24](#_Toc171166637)

[**5.1.1.** Relatia intre contractele inteligente ale sistemului de trasabilitate 24](#_Toc171166638)

[**5.1.2.** Definirea structurilor de date necesare pentru sistemul de trasabilitate 25](#_Toc171166639)

[**5.1.3.** Contractul inteligent „Lant de aprovizionare” - SupplyChain 28](#_Toc171166640)

[**5.1.4.** Contractul inteligent destinat pentru conturile actorilor - Users 29](#_Toc171166641)

[**5.1.5.** Contractul inteligent destinat administrarii produselor – Products 32](#_Toc171166642)

[5.2. Front-end: Interfata aplicatiei de trasabilitate 38](#_Toc171166643)

[**5.2.1.** Clase folosite pentru definirea de obiecte utilizate in comunicarea cu contractele inteligente – modulul models 38](#_Toc171166644)

[**5.2.2.** Clase utilizate pentru apelarea functiilor din contractele inteligente – modulul contracts 38](#_Toc171166645)

[**5.2.3.** Elemente de prezentare din interfata utilizator 40](#_Toc171166646)

[Capitolul 6. Testare și validare 42](#_Toc171166647)

[Capitolul 7. Manual de instalare si utilizare 44](#_Toc171166648)

[Capitolul 8. Concluzii 46](#_Toc171166649)

[Bibliografie 48](#_Toc171166650)

[Anexa 1 50](#_Toc171166651)

[Anexa 2 51](#_Toc171166652)

[Anexa 3 52](#_Toc171166653)

# Introducere

De-a lungul timpului s-au observat probleme in legatura cu natura alimentelor consumate, multiple accidente alimentare grave au aparut in diverse locuri ale lumii [1]. Consecintele au aparut asupra sanatatii publice, economiei si increderii consumatorilor. Problemele nerezolvate, atrag atentia si in prezent, avand in vedere ca in fiecare an, aproximativ 420.000 de persoane mor și alte 600 milioane se îmbolnăvesc din cauza consumului de alimente contaminate [2].

In contextul unei populații mondiale în creștere și al cerinței de a dubla producția de alimente până în 2050, lanțurile de aprovizionare agroalimentare se confruntă cu numeroase ineficiențe și probleme socio-economice și de mediu. [3]

Integritatea lanțurilor de aprovizionare agroalimentare este adesea compromisă prin incidente de siguranță alimentară rezultate din adulterare, diluare, modificare sau contrafacere. Frauda alimentară cauzează daune de până la 40 de miliarde de dolari anual [3]. Calitatea alimentelor influențează reputația și valoarea de piață a întreprinderilor, iar incidentele legate de siguranța alimentelor sunt tratate lent din cauza transparenței reduse și a sortării ineficiente a loturilor [4].

In anii 1990 trasabilitatea a fost introdusa in domeniile sanatatii, spatial si militar. Adoptia acesteia in industria alimentara a fost stimulata de cresterea numarului de pericole alimentare. Definitia trasabilitatii este: „*capacitatea de a urmări mișcarea unui furaj sau a unui aliment prin etapele specificate de producție, procesare și distribuție*” [5].

Principalele componente care formeaza trasabilitatatea sunt:

* Pastrarea inregistrarilor: permite colectarea datelor si recuperarea informatiilor necesare
* Urmarirea inapoi (tracing back) si urmarirea inainte (tracking forward): permit accesarea si verificarea informatiilor despre produse in orice etapa a lantului de aprovizionare

Trasabilitate nu garanteaza siguranta alimentara, dar in momentul in care este integrata cu alte sisteme de asigurare a calitatii, cum ar fi HACCP, ajuta la reducerea problemelor de natura alimentara [6].

Un sistem eficient de trasabilitate permite partenerilor din sistemul alimentar să urmărească produsele alimentare și ingredientele lor, de la recoltare până la procesare și, în final, până la piață [7]. O transparență și o trasabilitate mai bune pot răspunde cererii globale crescânde pentru produse agroalimentare obținute sustenabil și pot facilita identificarea și retragerea rapidă a loturilor contaminate în caz de contaminare alimentara [3].

Reglementările și standardele, inițiativele, mandatele clienților și obiectivele corporative sunt principalele influente pentru adoptarea tehnologiilor de diminuare a problemelor de natura alimentara. Reglementări precum Food Safety Modernization Act (FSMA) și inițiative precum New Era of Smarter Food Safety și Global Dialogue on Seafood Traceability au stimulat companiile să exploreze investiții tehnologice pentru a îmbunătăți trasabilitatea, siguranța alimentară și transparența lanțului de aprovizionare [7].

Cu toate ca majoritatea actorilor din lantul de aprovizionre sunt orientati spre maximizarea profiturilor, se observa o schimbare in ceea ce priveste transparenta si informarea consumatorilor pentru a putea face decizii in legatura cu consumul de produse. Studiile din [8] arata ca atat consumatorii, cat si participantii lantului de aprovizionare au o perceptie pozitiva asupra unui sistem de trasabilitate, evidentiind increderea, siguranta si sanatatea ca factori semnificativi.

Perceptiile consumatorilor sunt influentate de utilizarea unui sistem de trasabilitate care ofera transparenta. Spre exemplu, studiul din [2] ilustreaza cum etichetele de trasabilitate bazate pe blockchain au crescut perceptiile consumatorilor asupra calitatii produselor si, in consecinta, intentia de cumparare.

Astfel nevoia sistemelor de trasabilitate este data de reducerea posibilelor incidente alimentare, dorinta de a reduce fraudele si cererea consumatorilor de a oferi transparenta.

Sistemele de trasabilitate traditionale existente pana in acest moment sunt o solutie in ceea ce inseamna problemele alimentare. Cu toate astea, sistemele centralizate prezinta anumite vulnerabilitati si limitari:

* Lipsa de transparenta. Intr-un sistem centralizat, datele sunt controlate de o singura entitate sau grup restrans de entitati, ceea ce poate duce la lipsa de transparenta si la posibile manipulari ale datelor.
* Securitate limitata. Centralizarea datelor ceeaza un punct unic de esec care poate fi tinta atacurilor cibernetice, punand in pericol securitatea informatiilor.
* Costuri ridicate. Mentinerea si administrarea unui sistem centralizat poate fi costisitoare, implicand infrastructuri complexe si resurse umane dedicate.
* Eficienta scazuta. Verificarea datelor intre multiple entitati pot fi procese lente si laborioase.

Tehnologia blockchain ofera o solutie la aceste probleme prin descentralizarea si securizarea datelor intr-un mod transparent si imutabil. Blokchain-ul este un registru distribuit care inregistreaza tranzactii intr-un mod securizat si verificabil, eliminand nevoia de incredere intr-un singur punct central. Fiecare tranzactie este ciptografic sigilata si legata de tranzactiile anterioare, formand astfel un lant continuu de blocuri.

Ethereum, una dintre cele mai populare platforme blockchain, ofera functionalitati prin contractele inteligente (smart contracts), care permit automatizarea si securizarea proceselor complexe fara interventia intermediarilor. Aceste contracte inteligente sunt programe auto-executabile care ruleaza pe blockchain si pot gestiona tranzactii, stoca informatii si aplica reguli predefinite.

Implementarea unui sistem de trasabilitate pe blockchain-ul Ethereum aduce multiple avantaje:

* Transparenta. Toate tranzactiile si datele sunt vizibile pentru toti participantii la retea, asigurand transparenta si incredere.
* Imutabilitate. Odata inregistrate, datele nu pot fi modificate sau sterse, ceea ce garanteaza integritatea si autenticitatea informatiilor.
* Securitatea sporita. Datorita naturii distribuite si criptografiei, blockchain-ul este rezistent la atacuri cibernetice si la manipulari.
* Accesibilitate si inclusivitate. Toti participantii la lantul de aprovizionare au acces egal la date, ceea ce promoveaza o colaborare mai stransa si o distributie echitabila a informatiilor.

# Obiectivele proiectului

Tema propriu zisa consta in proiectarea si implementarea unui sistem de trasabilitate a produselor alimentare sub forma unei aplicatii distribuite care ofera utilizatorilor functionalitatile de baza necesare pentru monitorizarea produselor pe parcursul lantului de aprovizionare.

### Obiectivul principal

Obiectivul principal este de a oferi consumatorilor finali o modalitate de a verifica originea ingredientelor unui produs cumparat, asigurand ca datele acestuia sunt in conformitate cu realitatea. Realizarea obiectivului se va face cu utilizarea tehnologiei blockchain si a contractelor inteligente pentru a realiza un sistem independent, in care datele inregistrate vor fi imutabile pentru a asigura transparenta fata de consumator.

De asemenea, pentru atingerea obiectivului, sistemul este nevoit sa interactioneze cu actorii lantului de aprovizionare al produselor. Participantii trebuie sa fie capabili sa inregistreze modificarile din lantul de aprovizionare si sa ofere o imagine care sa reflecte realitatea. Sistemul trebuie sa fie capabil sa ofere o trasabilitate pe tot parcursul lantului de aprovizionare, si nu doar intre pasul anterior si cel imediat urmator.

### Obiective secundare

* Studiul solutiilor existente in blockchain pentru solutionarea problemei de trasabilitate in lantul de aprovizionare al produselor alimentare.
* Proiectarea si implementarea unor contracte inteligente care permit inregistrea produsele in blockchain.
* Proiectarea unor structuri de date pentru memorarea datelor.
* Implementarea unor algoritmi de trasabilitate.
* Evaluarea solutiei utilizate din punctul de vedere al costului operatiilor, dimensiunea datelor si nivelului de vulnerabilitati existente in solutie.
* Implementarea interfetei utilizator intuitive.

### Cerinte functionale

Cerintele functionale ale proiectului descriu functiile specifice pe care sistemul trebuie sa le execute pentru a indeplinii obiectivele principale si nevoile actorilor din lantul de aprovizionare, cat si nevoile consumatorilor. Sistemul are urmatoarele cerinte functionale:

1. Asigurarea ca datele sunt imutabile. Toate datele introduse in sistem trebuie sa nu poata fi modificate fara ca acest lucru sa fie semnalat. Aceasta cerinta asigura ca datele transmise intre actori sunt valide si ca ceea ce poate vizualiza consumatorul respecta realitatea si ii ofera incredere.
2. Inregistrarea de produse in lantul de trasabilitate. Sistemul trebuie sa permita actorilor de tip producator primar sa inregistreze date despre produsele pe care le creeaza. Datele trebuie sa includa informatii relevante pentru trasabilitate. Aceste inregistrari vor reprezenta punctul de plecare pentru produsele compuse si vor fi afisate ca noduri funza in interogarile privind trasabilitatea produselor compuse.
3. Inregistrarea procesului de fabricare a produselor. La fel ca produsele de baza, si produsele compuse trebuie sa fie inregistrate in sistem, dar de data aceasta functionalitatea nu presupune introducerea de date, ci utilizarea unor date deja disponibile pentru a se realiza transformarea unor produse din stoc. Prin aceasta actiune se dovedeste atat existenta unui nou produs, cat si utilizarea altor produse pentru realizarea acestuia.
4. Gestionarea produselor din stocul utilizatorului. Sistemul trebuie sa permita fiecarui participant să poată vizualiza stocul în timp real si sa gestioneze stocul de produse cu conditia ca modificarile efectuate asupra stocului sa fie monitorizate pentru a asigura transparenta fata de ceilalti participanti.
5. Gestionarea transferurilor de produse. Sistemul trebuie să permită urmărirea transferurilor de produse între utilizatori pentru a asigura transparența și pentru a actualiza stocurile atât ale expeditorului, cât și ale destinatarului. Fiecare transfer trebuie să fie înregistrat cu detalii precum data transferului, cantitatea transferată și identificarea clară a expeditorului și destinatarului.
6. Trasabilitatea unui produs. Sistemul va permite identificarea și urmărirea completă a oricărui produs prin intermediul unui cod de bare unic. Fiecare produs înregistrat în sistem va avea un cod de bare asociat, care va permite accesul la un istoric detaliat al tuturor transformărilor suferite de acesta și de ingredientele sale de la introducerea în sistem.

### Cerinte non-functionale

Cerintele non-functionale ale proiectului sunt menite sa asigure standarde de calitate si functionalitatea adecvata a sistemului pentru a satisface nevoile utilizatorilor. Sistemul prezinta urmatoarele cerinte de acest fel:

* Securitatea datelor. Sistemul trebuie sa asigure integritatea si autenticitatea datelor salvate in blockchain. Toate tranzactiile si modificarile intregistrate in blockchain trebuie sa fie securizate impotriva accesului neautorizat.
* Scalabilitate. Sistemul trebuie sa fie proiectat sa suporte o crestere a volumului de tranzactii si a numarului de utilizatori fara a compromite performanta si timpul de raspuns.
* Usurinta in utilizare. Interfata utilizator trebuie sa fie intuitiva si accesibila pentru persoane fara cunostinte tehnice avansate.

# Studiu bibliografic

## Sisteme de trasabilitate traditionale

Primul sistem de coduri de bare, brevetat în 1952, a fost adus la viață în anii 1970 datorită invenției laserului și miniaturizării progresive a calculatoarelor. Utilizarea codurilor de bare a generat date valoroase despre identificarea și mișcarea produselor, stimulând adoptarea rapidă a acestei tehnologii [7].

Sistemele de trasabilitate traditionale se bazeaza pe procese manuale si tehnologii simple, cum ar fi codurile de bare si documentele de transport. In aceste sisteme, produsele sunt etichetate cu coduri de bare care sunt scanate la diferite punct din lantul de aprovizionare, iar informatiile colectate sunt stocate intr-o baza de date centralizata. Aceste date sunt actualizate manual sau semi-automat in jurnale electronice, asigurand inregistrarea miscarii produselor. Documentele de transport contin informarii despre loturile de produse si ajuta la actualizarea registrelor centralizate la fiecare punct de control.

Cu toate acestea, sistemele traditionale se confrunta cu numeroase dezavantaje. Introducerea manuala a datelor creste riscul de erori umane, ceea ce poate afecta acuratetea si consistenta informatiilor. In plus, lipsa unei acatlizari in timp real a datelor poate duce la intarzieri in identificarea si rezolvarea problemelor, afectand eficienta generala a alantului de aprovizionare. De asemenea, transparenta este limitata, deoarece accesul la date este adesea restrictionat si distribuit intre diferiti actori ai lantului, facand dificila obtinerea unei viziuni de ansamblu complete. Securitatea datelor este un punct slab, deoarece documentele fizice si inregistrarile manuale sunt mai vulnerabile la manipulare si frauda [9].

Pe de alta parte, majoritatea dintre sistemele tradiționale de trasabilitate sunt centralizate, monopolizate, asimetrice și opace, ceea ce duce la crize de încredere [4]. In articolul [1] este mentrionat ca ca mai multe tari au adoptat numeroase reglementari pentru a spori masurile siguantei alimentare, dar arhitecturile folosite sunt centralizate sau distribuite, iar un atac asupra unui singur nod poate duce la falsificarea sau dezvaluirea datelor.

## Sisteme de trasabilitate bazate pe Blockchain

### Integrarea tehnologiei blockchain in pasii procesului de trasabilitate

Blockchain-ul îmbunătățește semnificativ fiecare pas din procesul de trasabilitate, oferind posibilitatea monitorizarii și verificarea fiecărei etape de la producție până la consumul final. Acest lucru nu numai că asigură conformitatea și calitatea produselor, dar și sporește încrederea consumatorilor prin oferirea de informații detaliate și verificabile despre produsele pe care le achiziționează .

Procesul de trasabilitate incepe cu fabricarea produselor, continua cu procesarea acestora, dupa care se ajunge la etapa de distributie si cea de vanzare. In etapa finala a trasabilitatii produsele ajung la consumator. Fiecare etapa este identificata si descrisa, evidentiind implementarile realizate in articolele studiate.

* Identificarea si inregistrarea initiala. Fiecare lot de produse sau materii prime este identificat unic și înregistrat pe blockchain. Aceasta poate include detalii precum data și locul de producție, materialele utilizate și specificațiile tehnice. Articolul [10] descrie cum fiecare lot de semințe este identificat prin coduri QR unice și ambalate la diferite niveluri, iar fermierii verifică autenticitatea semințelor prin scanarea acestor coduri QR.
* Monitorizare si inregistrare. În timpul procesării, toate etapele sunt monitorizate și înregistrate pe blockchain. Acest lucru include detalii despre fiecare etapă de procesare, mașinile utilizate, parametrii de procesare și personalul implicat. În articolul [11] procesarea include captarea datelor despre material, mașină, QA și post-procesare, care sunt documentate și stocate atât centralizat, cât și descentralizat .
* Controlul calitatii. Rezultatele testelor de calitate sunt, de asemenea, înregistrate pe blockchain, permițând urmărirea conformității produselor procesate cu specificațiile standard. Aceeași abordare este evidențiată și în articolul [12] unde rezultatele testelor de calitate sunt monitorizate prin senzori IoT și înregistrate în blockchain.
* Logistica si transportul. Datele despre transport, inclusiv timpul de tranzit, condițiile de transport (temperatură, umiditate etc.) și traseul urmat sunt stocate pe blockchain. Aceasta asigură transparența și permite monitorizarea în timp real a produselor în tranzit. Articolul [13] menționează utilizarea blockchain-ului pentru monitorizarea transportului în lanțurile de aprovizionare, asigurând că toate datele sunt vizibile pentru toți participanții din rețea .
* Verificarea autenticitaii la receptie. La fiecare punct de transfer sau recepție, produsele sunt verificate prin scanarea codurilor QR sau RFID, actualizând astfel blockchain-ul cu noi informații despre starea și locația produselor. Aceasta este o practică comună menționată și în [10] pentru verificarea autenticității produselor agricole pe parcursul lanțului de aprovizionare.
* Stocare si afisare. Retailerii pot utiliza blockchain-ul pentru a afișa detalii despre proveniența produselor, autenticitatea și istoricul procesării către consumatori, prin scanarea unui cod QR de pe ambalaj. Aceasta este o metodă eficientă menționată in [11] unde retailerii pot afișa detalii despre produse pentru consumatori.
* Gestiunea stocurilor. Stocurile sunt gestionate eficient, cu actualizări automate ale inventarului pe baza vânzărilor și aprovizionării în timp real, permițând astfel o planificare mai bună și reducerea pierderilor. Articolul [12] subliniază cum blockchain-ul poate facilita gestionarea inventarului prin actualizări automate și transparente .
* Transparenta si increderea. onsumatorii pot verifica detaliile complete ale produselor pe care le achiziționează, inclusiv originea, procesarea și transportul, crescând astfel încrederea în produse. Aceasta este evidențiată în articolul [13] care discută despre cum transparența blockchain-ului poate îmbunătăți încrederea consumatorilor.
* Siguranta produselor. În cazul unor rechemări sau probleme de siguranță, blockchain-ul permite identificarea rapidă și precisă a loturilor afectate, reducând astfel riscurile pentru consumatori și facilitând acțiunile corective rapide. Articolul [11] subliniază importanța acestui aspect în gestionarea rechemărilor și asigurarea siguranței produselor .

### Arhitecturi folosite in sistemele de trasabilitate

Aici pot sa spun despre arhitecturile folosite in general pentru a realiza un sistem, mai departe pot folosi aceste date ca referinta pentru implementarea mea

Hyperledger Fabric este o platformă care oferă soluții de registru distribuit, având o arhitectură modulară care asigură un grad ridicat de confidențialitate, reziliență, flexibilitate și scalabilitate. Aceasta elimină conceptul de mining, dar păstrează caracteristicile pozitive ale blockchain-urilor tipice, cum ar fi imutabilitatea blocurilor și determinismul ordinii evenimentelor. Arhitectura propusă în articolul [14] oferă un design inițial al unui sistem blockchain pentru trasabilitatea medicamentelor, identificând diferiți actori din lanțul de aprovizionare farmaceutic și stabilind relațiile dintre aceștia folosind canale diferite pentru a maximiza confidențialitatea și securitatea datelor.

Hyperledger Besu este o soluție de registru distribuit compatibilă cu Ethereum, care sprijină construirea de rețele private și integrarea cu blockchain-urile publice. Arhitectura permite trasabilitatea eficientă a medicamentelor, asigurând atât execuția privată a tranzacțiilor, cât și flexibilitatea arhitecturală și un debit ridicat al tranzacțiilor. Hyperledger Besu utilizează contracte inteligente și gestionarea cheilor private/publice pentru a semna și verifica tranzacțiile, menținând în același timp confidențialitatea și permisiunile complexe de gestionare​ [14].

Articolul [14] prezintă o comparație tehnică detaliată între platformele Ethereum, Hyperledger Fabric și Hyperledger Besu, evidențiind avantajele și dezavantajele fiecărei platforme. De exemplu, Hyperledger Fabric și Besu sunt orientate spre rețele private, permise, în timp ce Ethereum este mai orientat spre rețele publice. Fabric și Besu oferă o reconciliere mai rapidă a stării și o viteză superioară de execuție a tranzacțiilor.

Articolul [11] dezvoltă un concept de asigurare a calității pentru procesul de extrudare a materialului metalic, care permite o reprezentare digitală a lanțului valoric în forma unui registru de piese. Soluția descentralizată propusă constă dintr-o aplicație web pentru achiziția datelor, o soluție de stocare descentralizată pentru stocarea datelor de volum mare, un contract inteligent pentru captarea evenimentelor de fabricație și blockchain-ul Ethereum pentru stocarea transparentă și securizată a datelor.

Arhitectura propusă in [11] combină procesele fizice și digitale de fabricație, documentându-le într-un registru digital. Acesta include captarea datelor din procesele de dezvoltare, fabricație, sinterizare și control, stocarea acestora în baza de date MySQL și în sistemul descentralizat IPFS, și utilizarea unui contract inteligent pe blockchain-ul Ethereum pentru stocarea și accesul transparent la aceste date.

Solutia propusă a fost implementată și testată într-un studiu demonstrativ. Toate aplicațiile și procesele dezvoltate (aplicație web, stocare descentralizată, contract inteligent, acces la date prin Etherscan) au fost testate și validate în acest context.

### Studii de caz si aplicatii blockchain practice in sistemele de trasabilitate

Tehnologia blockchain ofera o solutie robusta pentru trasabilitatea alimentelor, asigurand imutabilitatea, transparenta si securitatea datelor. Blockchain permite inregistrarea tranzactiilor intr-un registru distribuit, care poate fi accesat de toate partile implicate in lantul de aprovizionare. Fiecare tranzactie este inregistrata intr-un registru distribuit care poate fi verificat de toti participantii, eliminand riscul de manipulare a datelor. Contractele inteligente automatizeaza si verifica tranzactiile pe baza unor conditii prestabilite.

In aricolul [15] se evalueaza patru proiecte blockchain pentru succesul lor in imbunatatirea trasabilitatii alimentelor organice. Primul proiect de studiu este Fairfood „Back to the Origin”, o solutie blockchain, open-source, publicata pentru a urmarii nucsoara de comert echitabil si organic de la ferma pana la consumator. Solutia ofera trasabilitate pentru multiple parti interesate. Blockchain-ul permite fermierilor sa confirme tranzactiile printr-o aplicatie pe dispozitivul mobil, asigurand astfel o trasabilitate completa si transparenta.

Al doilea proiect abordat, FishTales, urmareste tonul pescuit la undita de la pescari pana la consumator. Solutia a fost creata pentru a asigura trasabilitatea tonului de comert echitabil, dar proiectul nu a depasit faza conceptuala din cauza problemelor de adoptie si dificultatii in interfatarea cu procesele pe hartie existente.

Cel de al treilea proiect apartine unui retailer anonim si este orietat spre productia de fructe citrice in care blockchain-ul folosit este unul privat. Consumatorii pot scana un cod QR pe eticheta unui produs pentru a obtine informatii despre produs si trasabilitate. Solutia blockchain ofera transparenta si informatii detaliate despre originea produsului pentru consumatori.

Ultimul proiect evaluat in articol este realizat de un ONG Anonim, proiectul pilot blockchain public urmareste orezul organic si de comert echitabil de-a lungul lantului valoric. Desi proiectul nu a continuat, initiativa a urmarit sa asigure trasabilitatea completa de la ferma la consumator, concentrandu-se pe transparenta lantului de aprovizionare si pe validarea originii produselor.

Proiectele care au avut succes, Farifood si retailerul anonim, au demonstrat imbunatatiri semnificative in trasabilitatea pe intregul lant de aprovizionare, iar FishTales si ONG-ul Anonim, desi nu au avut succes, au fost proiecte care au evidentiat importanta transparentei intre partenerii lantului de aprovizionare.

Beneficiile evidentiate de proiectele care au avut succes sunt verificarea usoara a datelor de certificare, responsabilitatea crescuta in lantul de aprovizionare si simplificarea colectarii si schimbului de date.

O alta implementarea de succes o reprezinta Walmart, care a colaborat cu IBM și Universitatea Tsinghua pentru a crea un sistem de securitate alimentară bazat pe blockchain, vizând trasabilitatea și prevenirea fraudei alimentare. Într-un studiu pilot, timpul de urmărire a mango-urilor a scăzut de la 7 zile la 2,2 secunde folosind blockchain. Centrul de colaborare pentru siguranța alimentară din China permite clienților să acceseze informații detaliate despre produsele achiziționate, demonstrând autenticitatea acestora. Datorită succesului, în septembrie 2019, Walmart a impus furnizorilor de legume cu frunze să folosească blockchain, planificând extinderea sistemului și pentru alte produse proaspete [6].

Un sistem implementat pentru obtinerea trasabilitatii in industria de cafea, se poate observa in [8]. Autorii utilizeaza Polygon ca infrastructură blockchain. Blockchain-ul Polygon are avantaje în ceea ce privește compatibilitatea cu Ethereum, scalabilitatea ridicată, taxele de tranzacție reduse și ușurința în personalizarea aspectelor precum mecanismul de consens, dimensiunea blocurilor, timpul de generare a blocurilor și contractele inteligente configurabile. Aplicația este dezvoltată folosind Next.js, un framework pentru aplicații React, iar pentru baza de date, se utilizeaza MySQL.

Sistemul din [8] procesează aproximativ 10-20 de operațiuni paralele pe secundă, deși blockchain-ul Polygon pe care îl utilizează poate gestiona până la 65.000 de tranzacții pe secundă. Evaluarea rentabilității se face prin calcularea costului total pentru fiecare flux al lanțului de aprovizionare, inclusiv costurile de scriere și execuție. În timpul testării, taxele de tranzacție pe Polygon au variat între 0,0005 și 0,2 dolari. Evaluarea adoptării de către utilizatori, bazată pe chestionare, a arătat că 75% dintre părțile interesate consideră sistemul ușor de utilizat, dar necesită îmbunătățiri ale interfeței pentru a spori experiența utilizatorului și rata de adoptare.

### Integrarea blockchain-ului cu alte tehnologii

Articolul [16] evidentiaza integrarea tehnologiei blockchain cu IoT pentru a colecta si a intregistra automat datele despre produse. Sistemul propus este compus din trei componente pricipale, clientul, serverul si blokchain-ul. Responsabilitatea clientului este sa colecteze date prin dispozitive IoT. Serverul gestioneaza procesarea pentru logica de business, inclusiv managementul angajatilor si al dispozitivelor, parametrii sistemului si securitatea. Blockchain-ul asigura stocarea datelor, consensul, criptarea si decriptarea.

### Provocari si vulnerabilitati intalnite

Provocarile comune din toate proiectele analizate in articolul [15] sunt confidentialitatea datelor, adoptarea tehnologiei, validarea datelor si interoperabilitatea. Scalabilitatea este una dintre principalele probleme ale blockchain-ului. blockchain-urile tradiționale, precum Bitcoin și Ethereum, pot procesa un număr limitat de tranzacții pe secundă, ceea ce poate duce la întârzieri și la costuri mai mari pe măsură ce numărul de tranzacții crește. Aceasta provocare este intalnita in articolele [2] si [14].

De asemenea, si articolul [17] specifica ca scalabilitatea blockchain-ului este o provocare majoră, având în vedere dimensiunea vastă a lanțului global de aprovizionare alimentară. Sistemele blockchain alimentare nu necesită mineritul asociat cu criptomonedele, dar volumul de date generat poate fi semnificativ. Comparativ cu platformele globale de date individuale, lanțul de aprovizionare alimentară ar putea ajunge la scară de Petabytes pe an.

Autorii din [16] ajung la concluzia ca blockchain-ul are probleme legate de spatiul de stocare si rata de procesare a tranzactiilor, si considera disponibilitatea publica a continutului tranzactiilor ca fiind o problema de confidentialitate. Într-un sistem de trasabilitate, anumite informații pot fi sensibile și nu ar trebui să fie accesibile tuturor participanților. În articolul [11], se menționează necesitatea protejării datelor confidențiale în contextul unui registru distribuit.

Lipsa de interoperabilitate între diferitele platforme blockchain și sistemele tradiționale de trasabilitate poate reprezenta o problemă majoră. În articolul [10], se discută despre provocările legate de interoperabilitatea între diferitele soluții tehnologice existente în lanțurile de aprovizionare.

In articolul [1] sunt prezentate mai multe vulnerabilitati si tipuri de atacuri posibile, printre care:

1. **Vulnerabilități ale contractelor inteligente.** Contractele inteligente sunt susceptibile la erori de codare și vulnerabilități care pot fi exploatate de atacatori.
2. **Centralizarea relativă.** Deși blockchain-ul este o tehnologie descentralizată, implementările specifice pot suferi de centralizare relativă, unde un număr mic de noduri sau participanți controlează rețeaua.
3. **Atacul 51%.** Dacă un atacator sau un grup de atacatori controlează mai mult de 50% din puterea de calcul a rețelei blockchain, aceștia pot manipula blockchain-ul, inversa tranzacții și dubla cheltuieli.

### Concluzia rezultata pe baza articolelor in ceea ce priveste utilizarea tehnologiei blockchain in sistemele de trasabilitate

Problemele de scalabilitate, costurile de implementare și operare, complexitatea tehnică, interoperabilitatea și confidențialitatea datelor sunt aspecte care pot încetini adoptarea pe scară largă a blockchain-ului. De exemplu, scalabilitatea limitată a blockchain-urilor tradiționale poate duce la întârzieri și costuri ridicate pe măsură ce volumul tranzacțiilor creste [14]. Totuși, cercetările și dezvoltările continue în domeniul blockchain-ului sugerează că aceste probleme pot fi depășite în timp. Tehnologii emergente, cum ar fi algoritmii de consens mai eficienți și soluțiile de scalabilitate de nivel doi (e.g., sidechains, sharding), promit să îmbunătățească performanța și eficiența blockchain-ului​ [14].

În ciuda acestor provocări, avantajele unui sistem de trasabilitate descentralizat bazat pe blockchain depășesc clar dezavantajele. Un sistem descentralizat elimină necesitatea unei entități centralizate de încredere, reducând riscurile de manipulare a datelor și oferind un nivel ridicat de transparență și securitate. Toți participanții din lanțul de aprovizionare pot verifica și valida tranzacțiile, asigurând integritatea datelor și sporind încrederea în sistem [14] [13].

În concluzie, deși tehnologia blockchain întâmpină anumite probleme și dezavantaje, acestea sunt gestionabile și pot fi depășite prin inovații tehnologice și adoptarea de noi soluții. Soluția descentralizată oferită de blockchain este superioară unui sistem controlat de o singură entitate, asigurând un nivel mai mare de încredere, securitate și transparență pentru toți participanții din lanțurile de aprovizionare. Pe măsură ce tehnologia evoluează, blockchain-ul va juca un rol din ce în ce mai important în asigurarea trasabilității și conformității produselor în diverse industrii.

# Analiză și fundamentare Teoretică

## Concepte de baza in trasabilitate

Introducere / Descrie Trasabilitate

Exista trei concepte de baza care ajuta la succesul unui sistem de trasabilitate, iar acestea sunt: Elemente de date cheie, Evenimente critice de urmarire si Interoperabilitatea.

Elementele de date cheie sunt elemente de baza ale metadatelor necesare pentru a realiza cu succes cazurile de utilizare ale unui sistem de trasabilitate. Cu alte cuvinte aceste elemente sunt atributele inregistrate pentru a aborda domeniile de trasabilitate, cum ar fi autenticitatea si siguranta alimentara. Pentru trasabilitate infomatiile necesare sunt minime, datele necesare fiind: data de productie, data de expirare, cantitatea, expeditorul, destinatarul si identificatori.

Evenimentele critice de urmarire sunt puncte pe parcursul lantului de distributie unde elementele de date cheie trebuie sa fie transferate, transformate sau verificate. In general exista trei tipuri de evenimente: transport, transformare si epuizare. Transportul este transferul unui produs intre doua locatii si poate fi extern sau intern. Transformarea este un proces in care o schimbare substantiala si ireversibila se intampla unui produs si este necesara documentarea pentru a mentine trasabilitatea. Tranformarea are ca inputuri ingrediente si inregistreaza un output care captureaza producatorii, lotul, id-ul produsuluii astfel incat sa poata fi folosit in pasi ulteriori. Epuizarea depinde de consumarea sau aruncarea unui produs. In functie de produs, pot exista consideratii unice in ceea ce priveste cele mai bune practici pentru anumite evenimente critice de urmarire.

Interoperabilitatea

O mica concluzie

Avand in vedere ca acest proiect presupune implementarea unui singur sistem de trasabilitate, doar primele doua concepte vor fi utilizate.

## Interactiuni dintre participantii lantului de aprovizionare

## Cazuri de utilizare ale sistemului de trasabilitate

Cazurile de utilizare conceptualizeaza modul in care obiectivele sunt atinse prin actorii externi care utilizeaza sistemul. Pentru acest proiect am considerat implementarea unui sistem care indeplineste functiile de baza ale unui sistem de trasabilitate. In functie de nevoie producatorilor aceste functionalitati pot fi nevoite sa aiba o complexitate mai mare. Astfel, pentru un sistem de trasabilitate de baza, am stabilit urmaroarele cazuri de utilizare:

1. Introducerea produselor de materie prima. Acestea sunt punctele de plecare pentru orice produs care contine ingrediente, pot fi folosite de catre actorul care le-a introdus pentru creearea unui produs compus sau pot fi distribuite mai departe. In momentul introducerii detaliilor despre produs sunt colectate date despre cantitatea produsului, data de producere si data de expirare.
2. Crearea unui nou produs pe baza unei retete. Avand in vedere stocul utilizatorului, acesta poate crea un produs nou pe baza ingredientelor pe care le detine folosind o anumita reteta.
3. Vanzarea sau cumpararea unui produs. Acest pas presupune stabilirea cantitatii de produs ce se doreste a fi vandut sau cumparat, dupa care se creeaza o cerere de transfer pe care celalalt utilizator poate sa o accepte sau sa o respinga, iar in functie de statusul transferului, produsul poate sa fie scazut din stocul expeditorului si adaugat in stocul destinatarului.
4. Trasarea pasilor pentru creerea unui produs pe baza codului de bare. Procedura fundamentala a sistemului, verifica istoricul produsului si a ingredientelor acestuia, pana la produsele de baza. Ar trebui sa furnizeze date despre data la care ingredientele au fost create, data lor de expirare, cine le-a produs si codul de bare pentru identificarea lor.
5. Anularea unei operatiuni. Avand in vedere ca oamenii sunt utilizatorii acestui sistem, anumite greseli nedorite pot sa apara, astfel anularea unei creeri a unui produs sau a unui transfer se poate realiza cat timp evenimentul este ultimul creat si nu s-au creat alte evenimente care sa depinda de acesta.
6. Blocarea unui produs si retragerea acestuia. In momentul unei probleme alimentare si in urma identificarii cauzei acesteia, o autoritate sau un producator trebuie sa anunte toti utilizatorii care detin un produs infectat. Asfel ceilalti participanti nu vor utiliza produsul respectiv.

## Arhitectura aplicatiei

Aceasta aplicatie consta in crearea de contracte inteligente, programe care vor rula in blockchain, acestea comunica intre ele si permit conexiunea din exterior a utilizatorilor. Contractele inteligente definesc structuri de date, variabile care memoreaza date in blockchain si functii, toate fiind necesare pentru a dezvolta un sistem de trasabilitate intr-un lant de aprovizionare al alimentelor.

Pentru o usoara interactiune a utilizatorilor, este nevoie de o aplicatie client care apeleaza functiile din contractele inteligente. Aceasta aplicatie poate fi una desktop, web sau mobile, pentru acest proiect am ales dezvoltarea unei aplicatii web pentru ca poate fii accesata atat de pe un dispozitiv mobile, cat si de pe un laptop sau calculator.

Comunicarea dintre aplicatia client si contractele inteligente se poate realiza prin configurarea propriului nod care ruleaza software-ul Ethereum blockchain sau prin utilizarea unor noduri furnizate de servicii terte, cum ar fi Infura, Alchemy sau Quicknode.

Furnizorii (providers) sunt nodurile la care aplicatia client se conecteaza pentru a putea interactiona cu blockchain-ul, acestea pot fi noduri proprii sau noduri deja create de catre furnizori. Fiecare client Ethereum, furnizorii spre exemplu, implementeaza o specificatie JSON-RPC, un protocol de apel de procedura la distanta, care defineste mai multe structuri de date si regulile de procesare a acestora. Principalul avantaj al acestui protocol este ca poate folosii orice mod de transport al mesajelor, spre exemplu http sau sockets.

Odata ce s-a stabilit conexiunea la blockchain printr-un furnizor, aplicatia client poate citii starea salvata in blockchain. Citirea din blockchain nu necesita ca utilizatorul sa semneze tranzactia, dar daca acesta doreste sa creeze un nou produs folosind aplicatia client, aplicatia distribuita o sa ceara utilizatorului sa semneze tranzactia cu cheia sa privata, abia atunci aplicatia va transmite tranzactia spre blockchain, altfel tranzactia va fi respinsa de catre noduri.

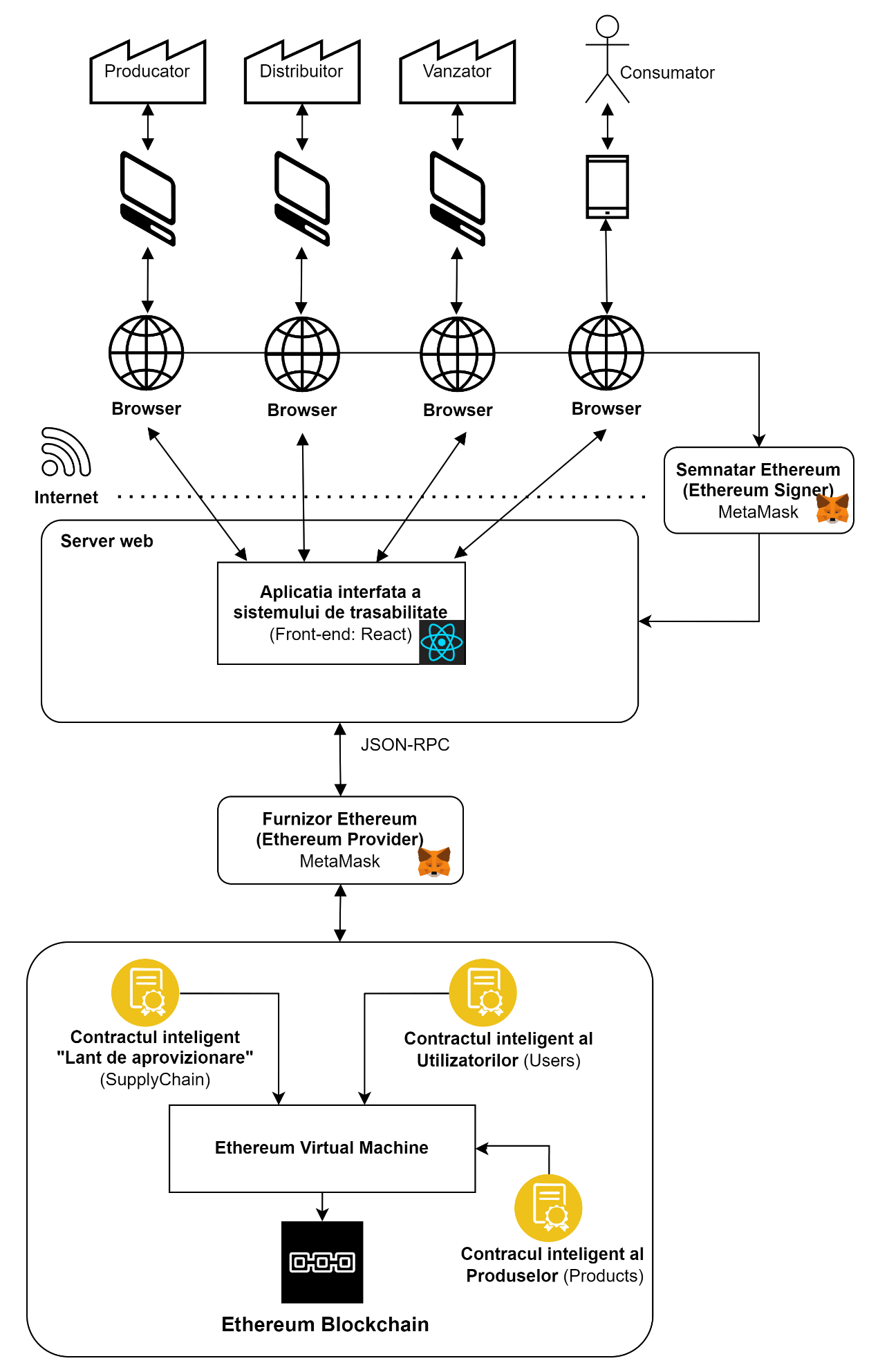


Figura 4.1 Diagrama conceptuala a proiectului

Pentru semnarea tranzactiilor se poate folosii instrumentul Metamask. Acesta salveaza cheia privata a utilizatorului in browser, si oricand aplicatia client are nevoie ca utilizatorul sa semneze o tranzactie, apeleaza la Metamask. De asemenea, instrumentul ofera si o conexiune la blockchain, in calitate de furnizor, deoarece are deja o conexiune la nodurile furnizate de Infura, de care are nevoie pentru a semna tranzactiile. Astfel, Metamask are un rol dublu, fiind atat un furnizor cat si un semnatar. Aacest lucru se poate observa si in Figura 4.8 in care este prezentata diagrama conceptuala a acestui proiect.

Avantajul acestei solutii fata de o solutie centralizata este controlul, care nu apartine unei institutii sau unei autoritati, ci acesta este oferit tuturor participantilor. Astfel, posibilitatea de a produce vreo frauda din punctul de vedere al informatiilor este redus, deoarece datele salvate intr-un blockchain sunt imutabile, adica nimeni nu le poate modifica o data ce acestea au fost salvate. De asemenea, transparenta tuturor tranzactiilor realizate in blockchain aduce incredere consumatorilor, acestia fiind capabili sa verifice fiecare tranzactie.

## Salvarea elementelor de date cheie in contractele inteligente

Un contract inteligent salveaza datele in blockchain prin scrierea si executarea codului sau pe reteaua blockchain, unde fiecare nod din retea valideaza si inregistreaza aceste tranzactii.

Codul contractului inteligent poate sa fie scris utilizant limbaje de programare specifice blockchain-ului respectiv, cum ar fi Solididty pentru Ethereum.

Mai departe contractul inteligent este implementat (deployed) pe blockchain, astfel codul contractului inteligent si datele initiale devin parte a lantului de blocuri odata ce tranzactia este confirmata.

Din momentul in care contractul este disponibil, utilizatorii pot sa trimita tranzactii pentru a apela functii din contractul inteligent. Aceste tranzactii sunt validatede nodurile retelei, mai exact fiecare nod executa codul contractului inteligent intr-un mediu virtual securizat (de exemplu, Ethereum Virtual Machine - EVM), pentru a se asigura ca regulile contractului sunt respectate si ca tranzactia este valida.

Odata ce o tranzactie este valida, rezultatul executiei este inregistrat intr-un nou bloc din blockchain. Datele stocate pot include orice tip de informatie definita in contract, de la stari simple, precum numere si siruri de caractere, pana la structuri de date mai complexe [18].

Urmatorul exemplu de contract permite unui utilizator sa salveze un intreg in variabila de stare produsSalvat si sa modifice sau sa obtina valoarea acesteia folosind functiile salvareProdus() sau obtinereProdus().

**/\* Definirea unui contract inteligent \*/**

contract ExempluStocareDate {

// **o variabila de stare de tip intreg fara semn**

uint produsSalvat;

**/\* Functie de salvare care poate modifica variabila de stare \*/**

function salvareProdus(uint produs) {

produsSalvat = produs;

}

**/\* Functie care returneaza valoarea variabilei de stare \*/**

function obtinereProdus() returns (uint) {

return produsSalvat;

}

}

Pe baza acestui exemplu se pot dezvolta contracte inteligente care sa salveze informatii despre orice element necesar unui sistem de trasabilitate. Se pot memora date despre actorii lantului de aprovizionare, produse, posibile transformari ale produselor si transferurile realizate intre participanti.

De asemenea, proiectarea contratelor inteligente trebuie sa ia in considerare costurile de tranzactie si limitarile gazului pentru a asigura ca functioarea contractului este viabila economic pe blockchain.

Datele esentiale pentru monitorizarea actiunilor realizate de actorii lantului de aprovizionare nu ar trebui sa ocupe foarte mult spatiu. Cu toate astea, in cazul in care structura de date ar creste si s-ar adauga detalii care nu ar fi triviale pentru functionarea sistemului ar trebui ca datele sa fie salvate intr-un sistem separat, nu in blockchain. Implementarea acestui lucru este posibila cu ajutorul protocolului IPFS, care permite stocarea si partajarea datelor intr-un mod distribuit.

## Etapa de transformare a materiilor prime in produse

Reteta este un element de date cheie care defineste procesul de transformare in fabricarea produselor. Primeste ca input un produs sau a mai mule produse intr-o cantitate bine stabilita si are ca output un anumit numar de produse, rezultate in urma transformarii. O reteta poate defini transformari sub forma unui blackbox, fara sa se urmareasca toate transformarile pe care ingredientele le sufera, sau poate fi folosita pentru a definii clar fiecare pas intermediar in procesul de productie. Aceasta functionalitate poate fi exploatata pentru a intalnii nevoile fiecarui producator pentru atingerea scopurilor si obiectivelor proprii.

Fiecare actor are un stoc de produse, acestea pot sa provina de la alti actori sau sa fie produse de baza in cazul in care actorul este si un producator primar. In functie de ce se doreste a fi creat, actorul poate creea un nou produs folosind o reteta, ingredientele disponibile in stocul utilizatorului sunt modificate, cantitatea acestora fiind scazuta, iar produsul nou rezultat in urma transformarii este introdus in stoc. Acest proces trebuie sa fie in conformitate cu realitatea pentru ca sistemul de trasabilitate sa fie unul de succes. Inainte de a se realiza orice modificare sistemul verifica daca produsele sunt in stoc si daca acestea sunt in cantitatea necesara pentru a se realiza transformarea. De asemenea o reteta nu poate fi modificata de un utilizator in momentul folosirii acesteia, retetele fiind bine stabilite dinainte, ele pot fi doar vizualizate si folosite pentru creearea altor produse.

Aplicatia este responsabila de crearea unor produse, unele introduse direct de catre utilizatori, altele pe baza unor retete, dar indiferent de felul de creare al unui produs, acesta primeste un cod de bare unic, generat in formatul EAN13.

Algoritmul de creare a unui produs pe baza unei retete, folosind ingrediente din stocul existent al producatorului presupune identificarea retetei care se doreste a fi folosita, aceasta poate fi transmisa ca parametru, sau se poate transmite doar id-ul unei retete. Pentru a simplifica necesitatea de a transmite date, s-a folosit transmiterea doar a id-ul retetei deoarece retetele ar trebui sa fie salvate in blockchain si sa fie usor de accesat, de asemenea se evita compararea retetei din functie cu cea care este deja salvata in blockchain.

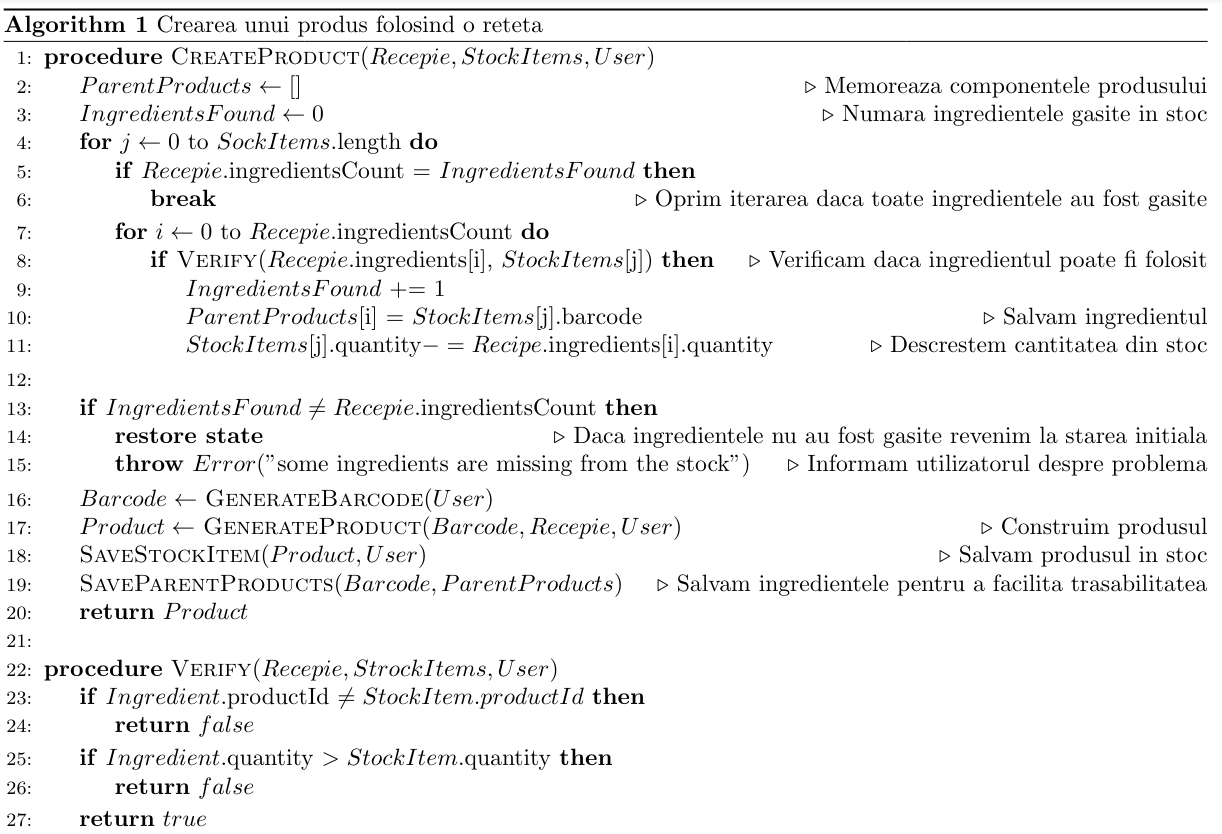


Figura 4.2 Algoritmul de creare unui produs pe baza unei retete

Dupa obtinerea retetei, urmeaza iterarea prin stocul curent al producatorului, unde pentru fiecare produs se face o iterare prin reteta in care se verifica daca produsul respecta cantitatea necesara specificata de ingredient, daca are acelasi tip de produs ca ingredientul si daca produsul nu a fost deja selectat anterior. Daca toate verificarile sunt indeplinite, se descreste cantitatea produsului din stoc cu cantitatea de ingredient folosita, se incrementeaza contorul ingredientelor gasite, si se salveza produsul ca fiind utilizat. Dupa toate parcurgerile se verifica contorul produselor gasite, daca acesta nu este egal cu numarul de ingrediente care este specificat in reteta, toate schimbarile realizate pana in acest moment sunt anulate. In caz contrar, daca totul continua cu succes, un produs nou este creat si salvat in blockchain. De asemenea produsul poate fi salvat in stocul utilizatorului, iar produsele care au fost marcate ca utilizate se pot stoca pentru a fi folosite in functionalitatea de trasabilitate.

Algoritmul de generare al codului de bare in format EAN13, are nevoie de trei parametrii de intrare, codul de regiune, codul producatorului si codul produsului. Se preiau ultimele 5 cifre din codul producatorului si se salveaza intr-o lista de 5 elemente, o sa o numim manufacturerDigits, de asemnea se salveaza intr-o alta lista, productDigits, ultimele 5 cifre din codul produsului.

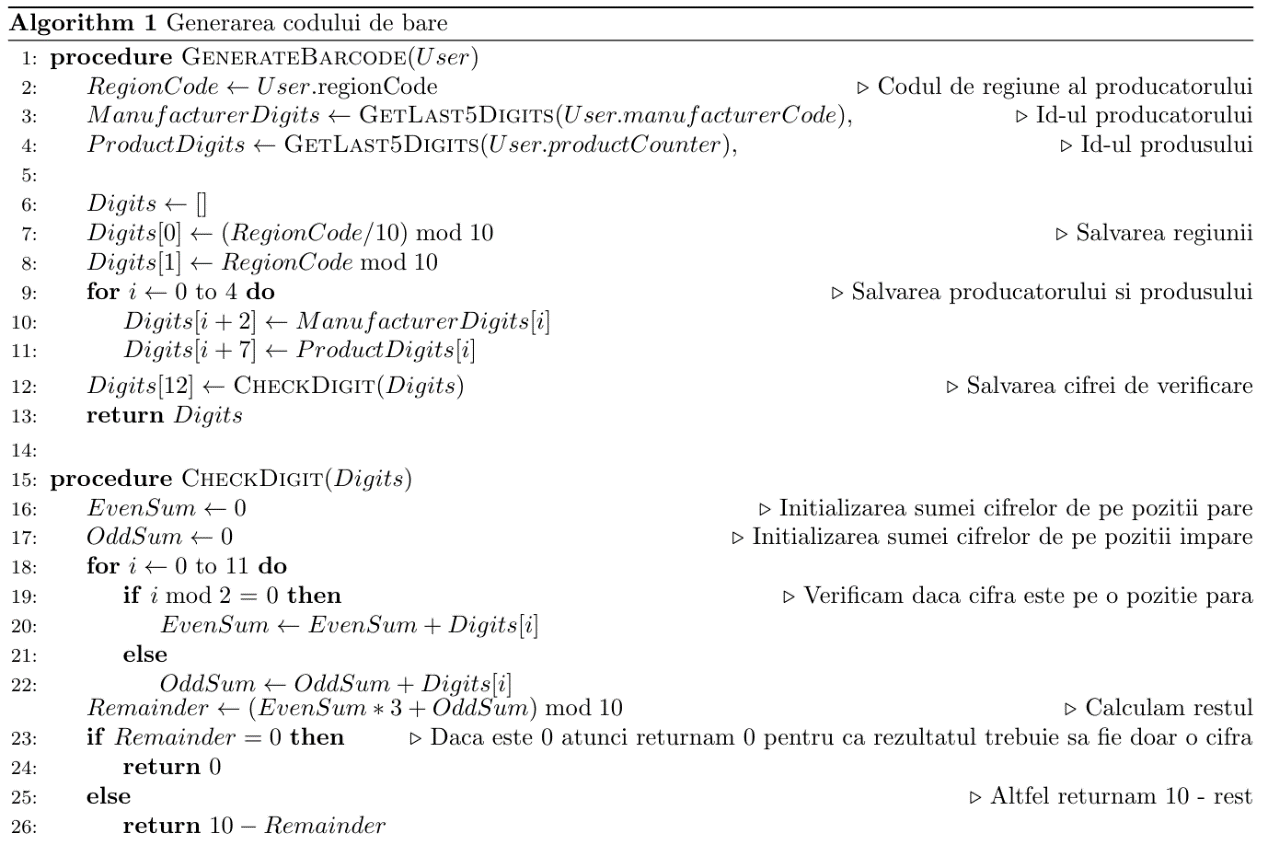


Figura 4.3 Algoritmul de generare al codului de bare

Mai departe se initializeaza o lista de cifre de lungime 13, digits. In primele doua pozitii ale listei digits se salveaza codul de regiune, care se preiau din primul parametru al functiei. Mai departe se face o parcurgere cu 5 iteratii care permite salvarea pe urmatoarele 5 pozitii din digits, adica pozitiile 2-6, valorillor din manufacturerDigits si pe urmatoarele 5, adica pozitiile 7-11, se salveaza valorile din productDigits. In final, pe pozitia 12 se salveaza cifra de control a codului de bare, acesta se genereaza pe baza cifrelor generate pana acum, se face suma elementelor de pe pozitie para, si suma elementelor de pe pozitia impara, dupa care se calculeaza o valoare cu formula (suma elementelor de pe pozitie para \* 3 + suma elementelor de pe pozitie impara) % 10, daca aceasta valoare este 0 se returneaza 0, altfel se returneaza diferenta dintre 10 si valoarea rezultata.

## Inregistrarea unui transfer de proprietate

Utilizatorul care doreste sa trimita un produs catre alt utilizator poate face asta prin initierea unui transfer. Transferul va aparea ca fiind in asteptare pentru utilizatorul destinatar, acesta fiind capabil sa accepte sau sa refuze transferul. Daca transferul este acceptat, atunci produsul este scazut din stocul expeditorului si este adaugat in stocul destinatarului, procesul se poate observa reprezentat in Figura 4.3. Pentru realizarea acestui proces se poate creea o lista de transferuri, unde ar fi salvate toate transferurile facute utilizatori. Utilizatorul destinatar ar initia transferul, astfel se adauga un transfer in lista, acesta avand statusul in asteptare, iar destinatarul poate accepta transferul, statusul schimbandu-se in acceptat, in momentul acceptari se realizeaza si modificarile stocurilor celor doi utilizatori. In momentul in care se refuza transferul, atunci statusul se schimba in refuzat si atunci nimic nu este modificat.

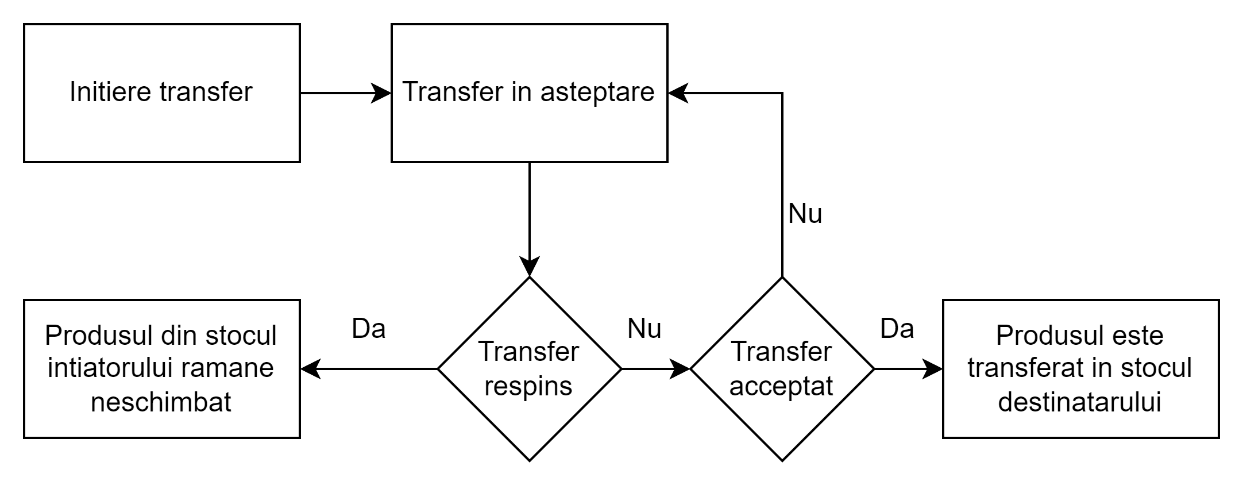


Figura 4.4 Transferul unui produs

## Functia de reconstructie a traseului unui produs

Traseul produselor este un proces care ofera detaliile unui produs intr-un mod amanuntit, aratand sub o forma arborescenta compozitia ingredientelor pana la produsele de baza care au fost folosite. Aceasta functionalitate poate fi obtinuta prin salvarea identificatorilor ingredientelor fiecarui produs care a fost creat vreodata. Cu aceasta abordare se poate face o cautare recursiva pana se intalnesc produsele de baza care au fost folosite.

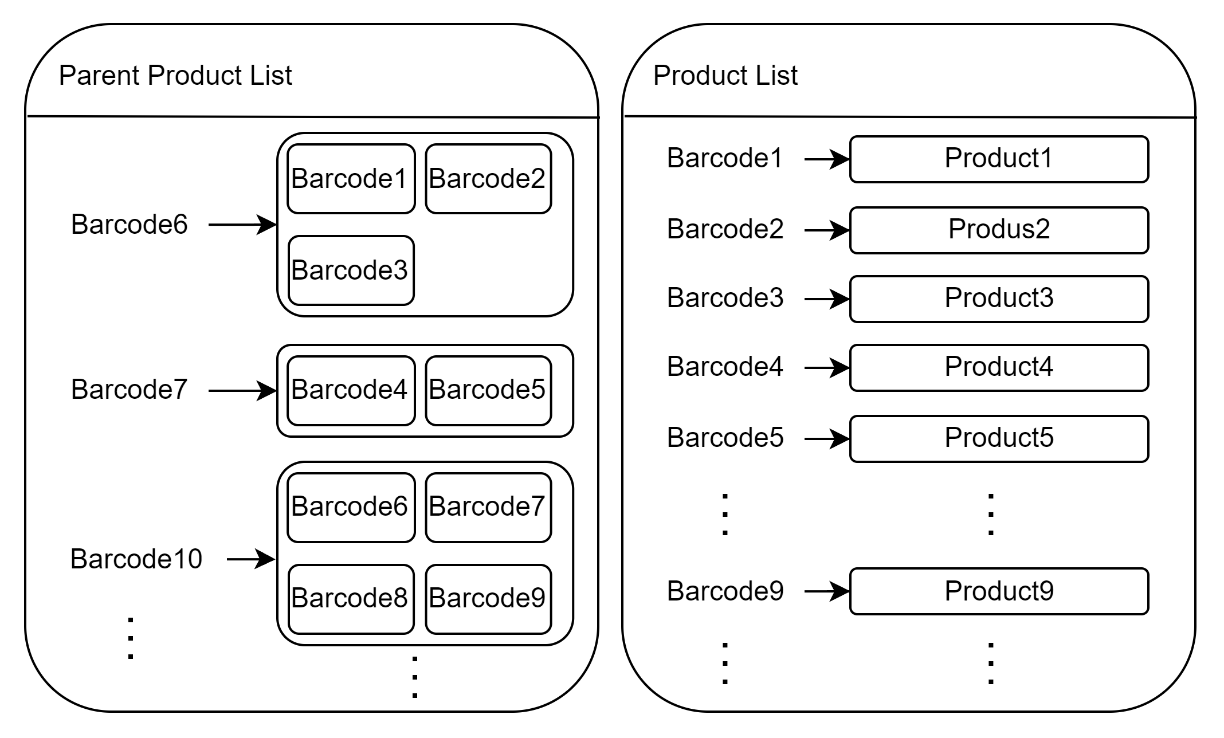


Figura 4.5 Structura de date a listei de produse si a listei de parinti

In primul rand, trebuie sa stabilim structura datelor salvate pe care urmeaza sa folosim algoritmul de restaurare a traseului unui produs. Structura datelor se poate observa in Figura 4.4 sub o reprezentare simpla. Produsele sunt salvate intr-o lista, acestea fiind identificate dupa codul lor de bare, iar parintii produsului sau ingredientele unui produs sunt date care sunt salvate separat de produse, acestea fiind stocate intr-o lista in care fiecare nod este identificat unic de codul de bare al produsului. Un element din aceasta lista contine o lista de coduri de bare care sunt identificatorii produselor ingrediente. Aceasta modalitate de stocare a parintilor este usor de gestionat si ocupa putina memorie, evitandu-se salvarea de obiecte mari sau de date irelevante.

Algoritmul foloseste structurile de date mentionate mai sus pentru a cauta informatii despre compozitia unui produs, ca intr-un final sa creeze o structura de date arborescenta cum se poate oberva in Figura 4.5. In aceasta structura un nod contine doua elemente produsul si o lista de noduri de acelasi tip care indica spre ingredientele produsului. Se poate observa ca frunzele unei astfel de structuri sunt noduri care contin doar produse, acestea sunt produse de baza care nu au fost formate din alte ingrediente si sunt componente din inceputul lantului de distributie.

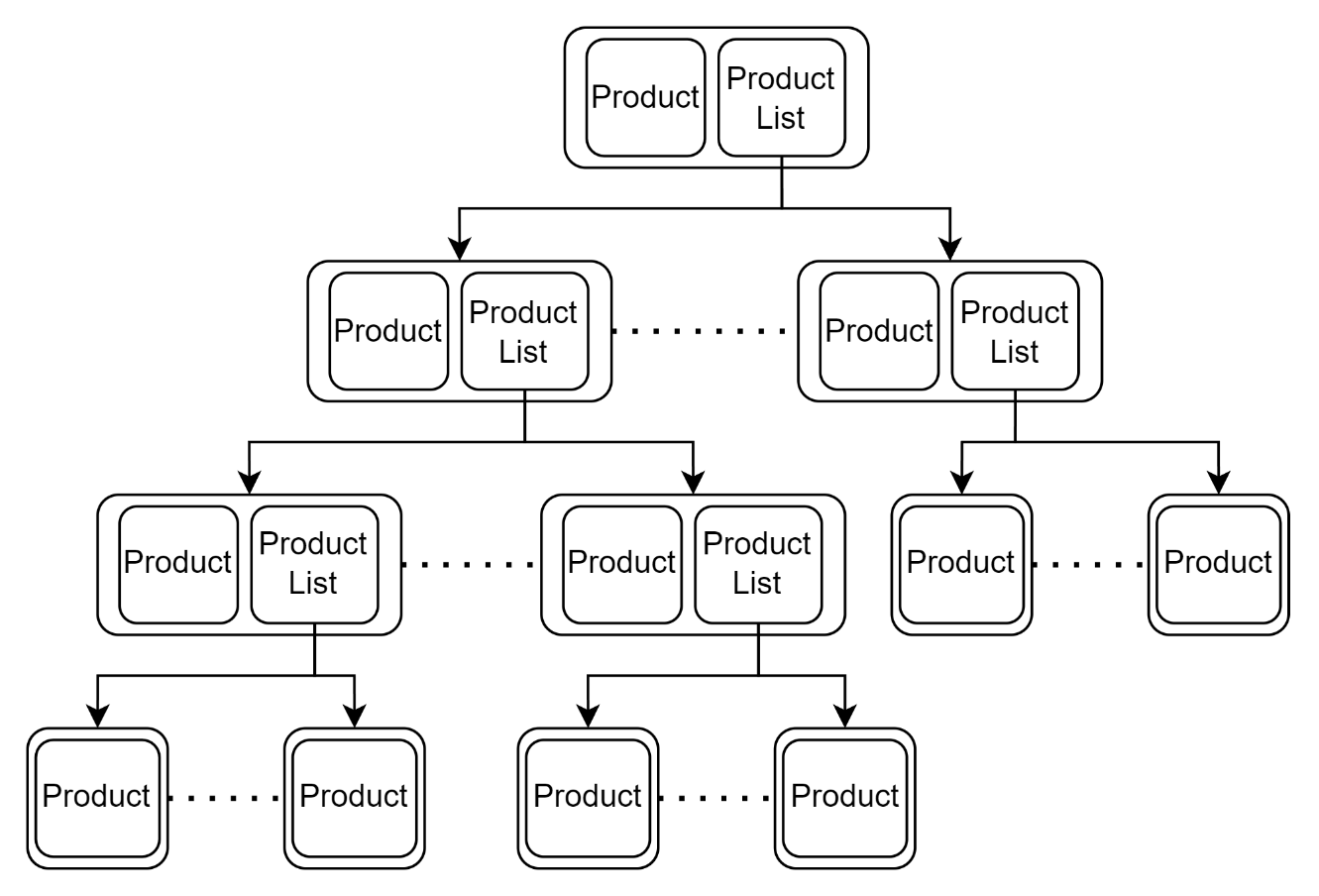


Figura 4.6 Arborele unui produs care ofera detalii despre compozitia acestuia

Refacerea traseului unui produs se realizeaza prin parcurgerea listei de parinti a produsului. In pseudocodul din Figura 4.6 am folosit functia GetProduct pentru accesarea un produs din lista de produse, si functia GetIngredientBarcode pentru accesarea listei de coduri de bare ale produselor parinte, respectiv ingredientele produsului, structurile acestor date au fost explicate anterior si pot fi vizualizare in Figura 4.4.

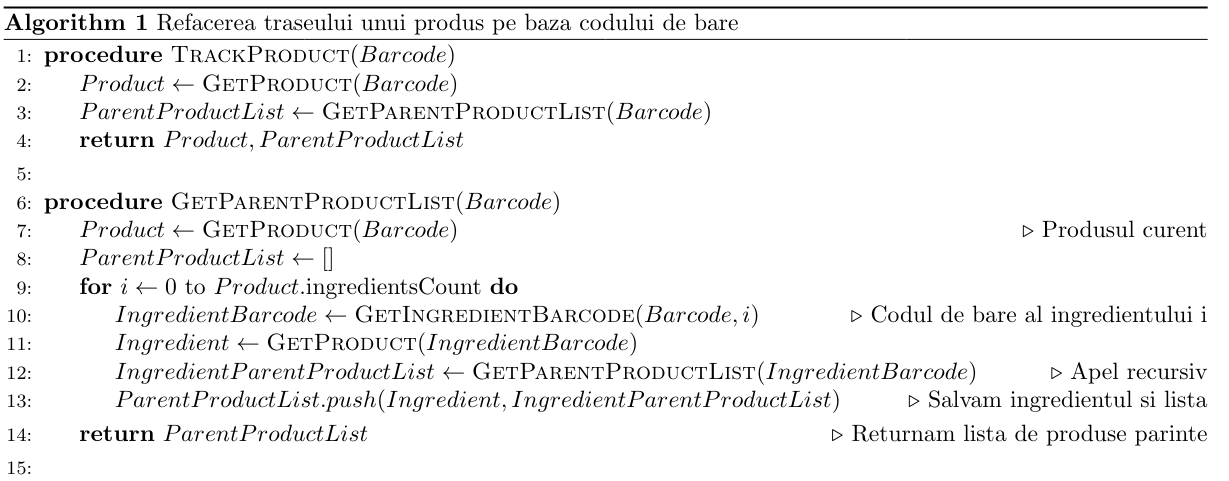


Figura 4.7 Algoritmul de refacere a traseului unui produs

Pasii algoritmului incep cu identificarea produsului de start dupa codul sau de bare si identificarea listei de coduri de bare a produselor parinte, dupa care urmeaza iterarea acestei liste, in timpur iteratiilor se identifica ingredientul accesand lista de produse. Ulterior, pentru a reusii sa cream structura arborescenta, este nevoie sa apelam recursiv functia pentru a salva lista de produse parinte a ingredientului. Mai departe, dupa revenirea din recursivitate, se salveaza ingredientul, impreuna cu lista sa de produse parinte intr-un nod, iar la final se returneaza lista de parinti a produsului. In final se va obtine un rezultat ca in Figura 4.5, aceasta strucura ne permite sa afisam datele intr-un mod usor de urmarit de catre utilizatori.

## Verificarile din cadrul sistemului de trasabilitate

Verificarile in cadrul utilizatorilor asigura ca functiile care se doresc sa fie utilizate sunt apelate de catre utilizatori reali. Daca un utilizator ar dorii sa faca actiuni in numele altui utilizator, acest lucru ar face ca operatiunea care se dorea sa sa efectueze sa fie anulata instant, astfel starea datelor raman nemodificate. Rolul unui utilizator este o alta verificare pe care o poate realiza sistemul de trasabilitate, aceasta fiind responsabila de autorizarea utilizatorilor in folosirea anumitor functii speciale, precum creearea unui produs. In ceea ce priveste trasferul unui produs de la un utilizator la altul, verificarea identitatii utilizatorului este necesara pentru acceptarea sau refuzul tranzactiei, daca aceasta verificare nu ar exista, atunci oricine ar putea refuza sau accepta un transfer in locul altui utilizator.

In cadrul produselor, verificarea joaca un rol important pentru trasabilitate. Spre exemplu, in momentul creari unui produs pe baza unor ingrediente, sistemul identifica ingredientele din stocul utilizatorului si verifica daca acestea sunt valabile, dupa care se verifica daca toate ingredientele au fost gasite. In cazul in care numarul de ingredientele gasite este mai mic decat numarul de ingrediente necesare, atunci sistemul renunta la creearea produsului si starea datelor revine la ceea initiala.

De asemenea, verificările făcute asupra datelor înainte de a ajunge în cadrul contractelor inteligente reduc erorile din sistem și procesarea datelor invalide, asigurând o funcționare mai eficientă și mai sigură a rețelei blockchain. Aceste verificări preliminare sunt esențiale pentru menținerea integrității și credibilității sistemului, evitând costurile și complicațiile asociate cu corectarea problemelor ulterior.

## Anularea unei actiuni

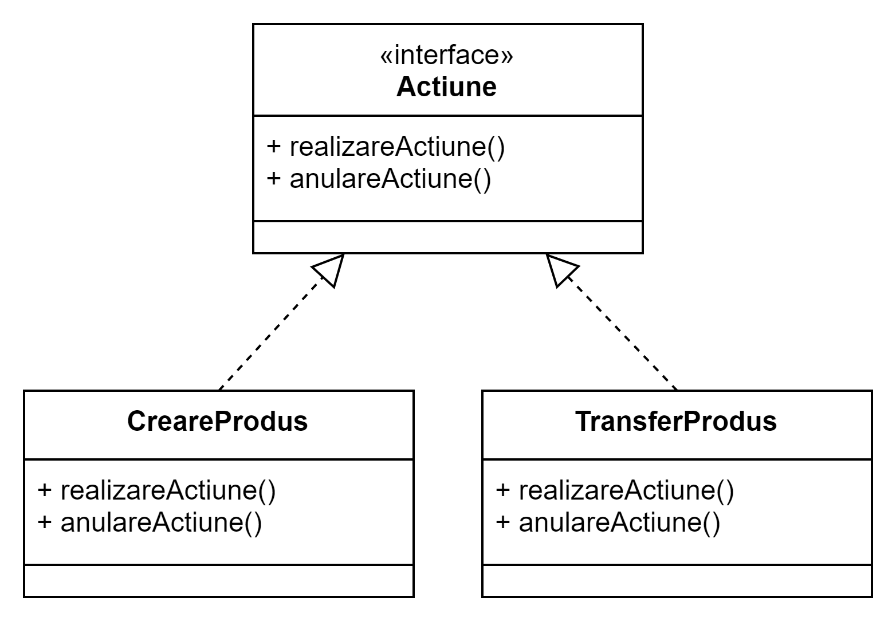
Anularea unei actiuni este o functionalitate care trebuie aplicata pentru fiecare actiune in parte, nu poate fi implementat ca ceva general. In cel mai fericit caz putem sa abordam acest subiect ca o implementare a unei interfete. Sa presupunem ca exista clasele CreareProdus si TransferProdus care implementeaza interfata Actiune, aceste clase au functionalitati diferite, dar putem spune ca acestea implementeaza functia realizareActiune() din interfata Actiune.

Figura 4.8 Reprezentarea implementarii interfetei „Actiune”

Folosind acelasi principiu, putem sa spunem ca anularea unei tranzactii presupune implementarea unei functii, anulareActiune() care trebuie sa trateze anularea unei actiuni in functie de ce a fost realizat in cadrul functiei care a modificat starea sistemului.

Exemple de anulari ale actiunilor din sistemul de trasabilitate:

1. Anularea inregistrarii ultimului produs introdus in sistem. Se verifica daca produsul a fost folosit in realizarea altor produse sau daca a fost folosit pentru realizarea unui transfer. Daca totul este in regula, adica produsul nu a suferit nici o modificare, atunci anularea se poate face cu succes.
2. Anularea initierii unui transfer. Daca s-a initializat un transfer spre un utilizator, iar intre timp expeditorul se razgandeste, atunci acesta poate anula transferul cat timp destinatarul nu l-a acceptat.

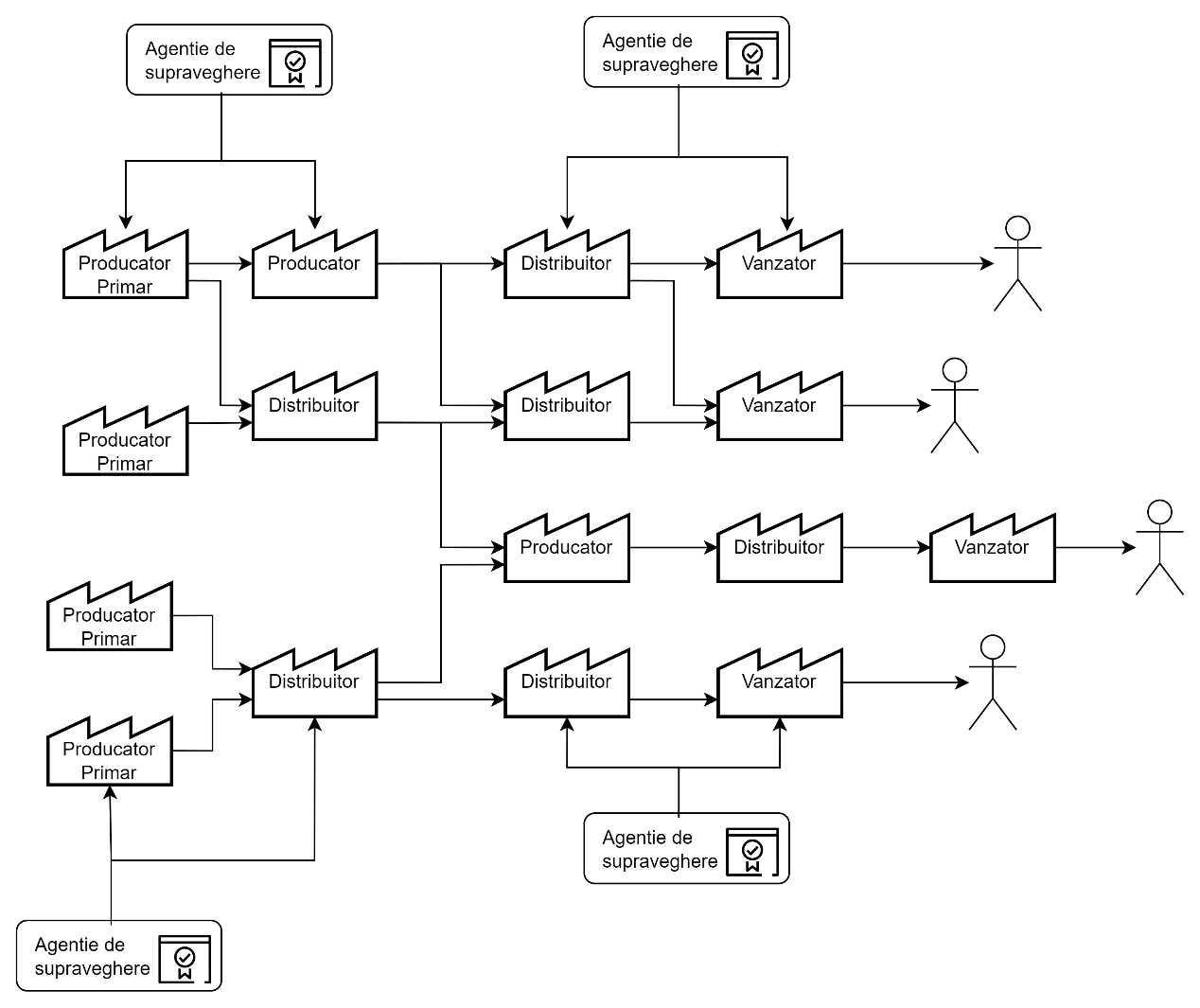
## Semnalarea unui produs contaminat in sistemul de trasabilitate

In momentul in care se detecteaza o neregula in legatura cu vreun produs, o autoritate poate sa atraga atentia tuturor celorlalti actori ai sistemului si sa blocheze produsul din a mai fi utilizat. Spre exemplu, daca o autoritate blocheaza produsul cu codul „1”, iar un producator doreste sa creeze un produs pe baza acestuia sau sa il transfere altei entitati, acest lucru nu va fi posibil, pentru ca sistemul va identifica ca produsul cu codul „1” este pe lista neagra si producatorul ar trebui sa scoata produsul din utilizare.

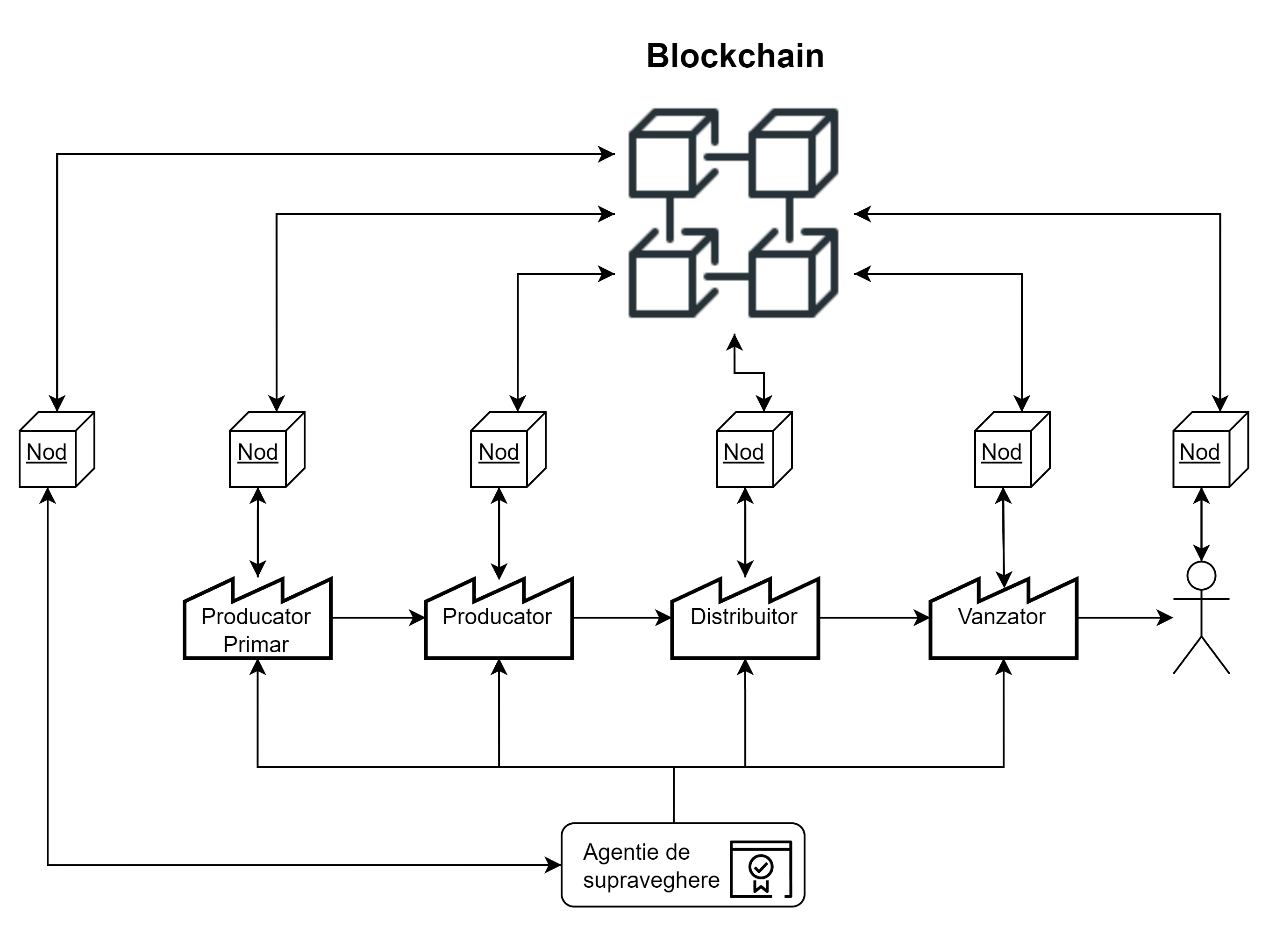
Aceasta functionalitate ar trebui sa fie una necesara, in care sistemul memoreaza o lista de coduri de bare in care sunt introduse produsele contaminate sau cu defecte, iar inainte de momentul utilizarii vreunui produs, sistemul ar trebui sa verifice ca acel produs sa nu fie in lista respectiva, in caz contrar orice operatiune ar trebui anulata si starea sistemului sa nu fie modificata.

Use this:

Actorii lantului de aprovizionare realizeaza transferuri intre ei



Also use this



# Proiectare de detaliu și implementare

## Back-end: Contractele inteligente ale sistemului de trasabilitate

### Relatia intre contractele inteligente ale sistemului de trasabilitate

Initial am creat un singur contract inteligent SupplyChain, in care aveam date despre produse si utilizatori. In timp ce aplicatia a crescut, am impartit contractul in trei parti, SupplyChain, Users si Products, care erau conectate folosind o relatie de mostenire, SuplyChain extinzand atat Users, cat si Products.

Din punct de vedere al codului, proiectul era usor de gestionat, dar dupa un timp, din cauza cresterii complexitatii, in momentul compilarii dimensiunea contractului inteligent a devenit prea mare, ceea ce era un impediment. Acest ultim aspect m-a facut sa folosesc conceptul de compozitie si cel de interfata, care poate fi observat in Figura 5.1.

Contractele inteligente Users si Products implementeaza interfetele UsersInterface, respectiv ProductsInterface, iar contractul inteligent SupplyChain contine doua obiecte, users (UsersInterface) si products (ProductsInterface). Legatura intre interfete si contractele inteligente se face in constructorul din SupplyChain care primeste ca parametrii adresele contractelor inteligente si le transmite mai departe, in momentul initializarii obiectelor users si products, spre constructorii interfetelor, ceea ce permite utilizarea contractelor inteligente Users si Products.

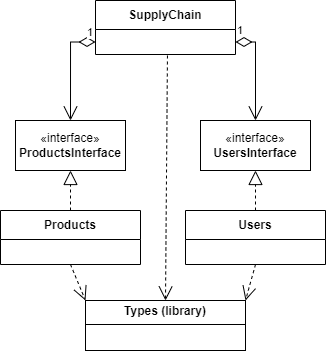


Figura 5.1 Diagrama UML a contractelor inteligente

### Definirea structurilor de date necesare pentru sistemul de trasabilitate

Biblioteca Types este un tip special de contract inteligent care este utilizat in toate celelalte contracte inteligente. Aceasta defineste majoritatea structurilor de date care sunt folosite in sistemul de trasabilitate.

#### Structuri de date pentru definirea actorilor din lanul de distributie

Rolurile actorilor din lantul de distributie sunt definite in structura de date de tip enumerare UserRole:

* Admin – administrator sau autoritate
* Manufacturer – producator, poate fi unul primar sau unul care sa produca produse pe baza altor produse
* Supplier – distribuitor de produse
* Vendor – vanzator de produse
* Customer – consumatorul unui produs, clientul final din lantul de distributie

Aceasta structura de date stabileste permisiunile pe care le are un anumit participant in cadrul sistemului de trasabilitate.

UserDetails este o structura folosita pentru a memora date despre actorii inregistrati in sistem, aceasta contine urmatoarele campuri:

* id – identificatorul actorului, adresa publica a portofelului digital al utilizatorului, este un obiect de tip address
* name – numele actorului, este un obiect de tip string
* email – email-ul de contact al actorului, este un obiect de tip string
* role – rolul actorului, este de tipul UserRole, care a fost descris anterior

Pentru a memora date specifice pentru un producator, am creat structura de date ManufactuerDetails:

* code – codul asignat unui producator, este de tip uint
* region – regiunea producatorului, este de tip uint

Aceasta structura va fi folosita in momentul inregistrarii unui produs in sistem, in special pentru generarea codului de bare al produsului.

#### Structuri de date pentru gestionarea produselor in sistemul de trasabilitate

Structura care defineste tipul de produs este ProductType, aceasta contine:

* id – identificator unic al tipului de produs, este de tip uint
* name – denumirea tipului de produs, este de tip string
* details – element de tip string, in care sunt salvate alte informatii, spre exemplu: compozitie chimica, descrierea aspectului

Obiectele care folosesc acest tip de date definesc ce produse pot fi create si reduce cantitatea de informatie din obiectul unui produs.

Pentru a adauga un obiect de tipul ProductType in sistemul de trasabilitate, am definit structura de date de tip transfer ProductTypeAddDTO:

* name – denumirea tipului de produs, este de tip string
* details – detalii, este de tip string

Folosirea structurii ProductTypeAddDTO in locul structurii ProductType este necesara pentru a evita transferul inutil de date, identificatorul obiectelor de tip ProductType fiind stabilit in contractul inteligent.

Elementul principal al proiectului este reprezentat de structura de date Product, care contine datele unui produs iserat in sistemul de trasabilitate de catre un producator. Structura contine urmatoarele informatii:

* name – denumirea produsului, este un obiect de tip string
* barcodeId – codul de bare al produsului este un element de tip string, unic pentru fiecare produs creat, ajuta la identificarea produselor in orice punct al sistemului
* productTypeId – identificatorul tipului de produs, este de tip uint, acesta leaga produsul de tipul sau de produs
* recipeId – element de tip uint, este identificatorul retei care a creat produsul, leaga produsul de reteta folosita in crearea acestuia
* ingredientsCount – numarul de ingrediente pe care produsul il contine, este de tip uint
* manufacturerId – identificatorul producatorului, este un obiect de tip address, acesta este necesar pentru a stabili trasabilitatea
* manufacturerName – numele producatorului, este un obiect de tip string
* manufacturingDate – data de producere a produsului, este un obiect de tip uint si este reprezentata in unix timestamp
* expirationDate – data de expirare a produsului, este un obiect de tip uint si este reprezentata in unix timestamp

In momentul introducerii unui obiect nou, fara ca acesta sa fie creat pe baza altor produse, se va folosi structura de date ProductAddDTO:

* productTypeId – identificatorul tipului de produs, obiect de tip uint
* manufacturingDate – data de producere a produsului, obiect de tip uint
* expirationDate – data de expirare a produsului, obiect de tip uint

Pentru definirea unei retete care exprima transforamrea produselor in alte produss, s-a folosit structura de date Recipe:

* id – identificatorul retetei, este un obiect de tip uint
* resultTypeId – tipul de produs care va rezulta, referinta spre un obiect de tip ProductType, este un obiect de tip uint
* resultTypeName – denumirea produsului rezultat, un obiect de tip string
* quantityResult – un obiect de tip uint care specifica numarul de produse care vor rezulta in urma transformarii

Pentru a usura crearea unei retete si a imbunatatii utilizarea memoriei, am creat structura care reprezinta un ingredient folosit intr-o reteta. Aceasta are denumirea RecipeIngredient, si are urmatoarele proprietati:

* recipeId – un obiect de tip uint, identificatorul retetei, acesta permite fiecarui ingredient sa stie din ce reteta face parte
* productTypeId – un obiect de tip uint, identificatorul tipului de produs al ingredientului
* productQuantity – un obiect de tip uint, specifica cantitatea de produs ingredient necesar in reteta

Structura care reprezinta un produs prezent in stocul unui utilizator este StockItem, aceasta determina produsul si cantitatea in care este disponibil:

* barcodeId – codul de bare al produsului, un obiect de tip string
* quantity – numarul de produse in stoc, un obiect de tip uint

#### Structuri de date pentru monitorizarea comercializarii de produs intre actorii sistemului de trasabilitate

Structura de date care identifica statusul unui transfer de produse este ObjectStatus, aceasta este o enumerare ce contine urmatoarele posibile valori:

* Pending – transferul este in asteptare
* Accepted – transferul a fost acceptat
* Refused – transferul a fost refuzat.

Structura care contine datele unui transfer de produs intre doi actori ai lantului de distributie este Transfer, aceasta contine:

* id – identificatorul transferului, obiect de tip uint
* sender – identificatorul actorului care a initiat transferul, expeditorul, un obiect de tip address
* receiver – identificatorul actorului destinatar, un obiect de tip address
* barcodeId – codul de bare al produsului care se doreste a se transfera, un obiect de tip string
* status – statusul transferului, un obiect de tip ObjectStatus, structura definita anterior

Relatiile principale dintre structurile de date mentionate mai sus, pot fi vizualizate in Figura 5.2 pentru o intelere mai usoara in ceea ce priveste folosirea acestora pe parcursul proiectului. Se observa ca elementul central este Product, acesta avand o legatura cu aproape toate structurile din reprezentarea grafica.

Transferul este legat de Product si de UserDetails, un utilizator poate crea si primi mai multe transferuri, iar un singur produs poate fi folosit in mai multe transferuri. Relatia dintre StockItem si Product este una many-to-one, fiind posibila existenta mai multor obiecte de tip StockItem care sa contina acelasi produs. Recipe este intr-o relatie de many-to-one cu Product, o reteta fiind folosita pentru crearea mai multor produse. ProductType este intr-o relatie tot one-to-many cu Product pentru ca un tip de produs poate fi intalnit in multe produse. In final, RecipeIngredient este intr-o relatie many-to-one atat cu ProductType, cat si cu Recipe, fiind posibila crearea mai multor obiecte de tipul RecipeIngredient care sa foloseasca acelasi tip de produs sau sa fie prezente intr-o reteta.



Figura 5.2 Diagrama conceptuala care ilustreaza relatiile dintre principalele structuri de date din sistemul de trasabilitate

### Contractul inteligent „Lant de aprovizionare” - SupplyChain

Dupa cum am specificat si mai sus, la punctul 5.1.1, contractul inteligent SupplyChain face legatura dintre contractele inteligente Users si Products. Ceea ce inseamna ca acesta implementeaza functii care apeleaza alte functii din contractele inteligente Users si Products. Cu toatea acestea, inainte de a realiza apeluri aprofundate se pot face verificari la acest nivel pentru a se evita consumul de resurse.

#### Functii pentru gestionarea conturilor actorilor din lantul de distributie

Functia de inserare a unui cont de utilizator este addUser. Aceasta primeste ca parametru un obiect care reprezinta detaliile actorului adaugat in sistem, user\_ (UserDetails) si apeleaza mai departe functia \_addUser, din contractul inteligent Users, cu doi parametrii, primul fiind obiectul user\_ si al doilea msg.sender care contine detalii despre contul de Ethereum care a apelat aceasta functie.

Functia de obtinere a unui cont de utilizator, getUser, primeste ca parametru identificatorul unui cont, userAddres (address), si returneaza un obiect care contine detaliile contului, un obiect de tip UserDetails care este rezultatul apelului functiei get din contractul inteligent Users.

Modificatorul este o functie speciala ce poate fi atribuita unei alte functii pentru a impune restrictii. In acest contract inteligent este definit un modificator care permite apelul unor functii sa fie facut doar de un producator. Modificatorul este onlyManufactuer, iar acesta verifica daca apelul functiei s-a facut de catre cineva cu o adresa de Ethereum valida si daca contul identificat de aceasta adresa are rolul UserRole.Manufacturer. In caz de eroare orice apel de functie va fi intrerupt.

#### Functii pentru gestionarea produselor din sistemul de trasabilitate

Functia folosita pentru adaugarea unui produs in sistem este addProduct, aceasta primeste ca parametru un obiect care reprezinta un produs, product\_(ProductAddDTO), si apeleaza mai departe functia \_addProduct din contractul inteligent Products. In implementarea acestei functii se poate observa cum SupplyChain poate evita o apelare inutila a functiei din contractul inteligent Products pentru ca in descrierea functiei addProducts este utilizat modificatorul onlyManufacturer.

Functia pentru adaugarea tipurilor de produse este addProductType, aceasta primeste ca parametru un obiect de tipul ProductTypeAddDTO si apeleaza mai departe functia \_addProductType din contractul inteligent Products pentru a se adauga un nou tip de produs in sistem.

Functia pentru crearea unui transfer intre doi utilizatori este createSellRequest. Functia primeste ca parametrii adresa cumparatorului, buyerId (address), codul de bare al produsului ce se doreste a fi transferat, barcodeId (string), timpul curent al initiatorului, currentTime(uint) si cantitatea de produs ce se va transfera, quantity(uint). Pentru a prelua datele conturilor de utilizator se apeleaza de doua ori functia get din contractul inteligent Users cu parametrii identificatorii conturilor, dupa care rezultatele sunt salvatate in obiectele buyer si seller, de tip UserDetails. Mai departe urmeaza sa se salveze un obiect de tip Transfer in sistemul de trasabilitate, pentru a obtine acest luctu se apeleaza functia \_createSellRequest din contractul inteligent Products care primeste ca parametrii barcodeId, buyer, seller, currentTime si quantity.

Functia pentru acceptarea sau refuzarea unui transfer este acceptSellRequest. Aceasta functie este apelata de catre un actor destinatar/cumparator.

Parametrii functiei sunt adresa vanzatorului, sellerId(address), codul de bare al produsului procesat pentru transfer, barcodeId(string), timpul curent, currentTime(uint) si raspunsul destinatarului la transferul inititat, acceptSell(bool). La fel ca in functia createSellRequest, datele utilizatorilor participanti sunt obtinute cu ajutorul functiei get din contractul inteligent Users si salvate in obiectele buyer si seller, dar de data aceasta participantii au rolurile inversate. Pentru finalizarea actiunii se apeleaza functia \_acceptSellRequest din contractul inteligent Products.

Functia folosita pentru crearea unui produs pe baza unei retete este createProduct. Identificarea retetei se realizeaza prin identificatorul primit ca parametru, recipeId(uint). In corpul functiei se identifica contul de utilizator care a apelat functia prin intermediul functiei get din contractul inteligent Users. De asemenea se preiau datele de producator cu ajutorul functiei getManufacturerDetails, tot din Users, dupa care se apeleaza mai departe functia \_createProduct din contractul inteligent Products pentru a se finaliza crearea produsului.

Pe de alta parte, aceasta functie se foloseste de modificatorul onlyManufacturer pentru a nu permite unor actori ce nu au rolul de producator sa foloseasca functia.

### Contractul inteligent destinat pentru conturile actorilor - Users

Contractul inteligent Users stocheaza informatii legate de actorii lantului de distributie si implementeaza functii pentru adaugarea si preluarea datelor. Relatiile dintre functiile si obiectele din contractul inteligent pot fi urmarite si in Figura 5.3, diagrama care prezinta interactiunea dintre functii si obiectele contractului inteligent.

In constructorul contractului se initializeaza utilizatorul initial admin\_, care are rolul de administrator, primeste numele si email-ul din parametrii constructorului, iar adresa lui este adresa utilizatorului care publica contractul inteligent in reteaua blockchain.

#### Obiecte declarate pentru memorarea datelor actorilor din sistemul de trasabilitate

* usersCount – obiect de tip uint, memoreaza numarul de utilizatori care au fost creati pana in acest moment
* manufacturersCount, suppliersCount, vendorsCount si customersCount – obiecte de tip uint, sunt de asemenea contoare pentru conturi de tipul Manufacturer, Supplier, Vendor si Customer, aceste contoare vor fi folosite in indexarea unor liste
* users – este un obiect de tip mapping(address => Types.UserDetails), memoreaza detaliile tututor utilizatorilor si poate fi accesat pe baza adresei acestora
* manufacturersList, suppliersList, vendorsList si customersList – sunt obiecte de tip mapping(uint => address), folosite pentru a separa conturilor pe roluri
* manufacturerDetailList – este un obiect de tip mapping(address => Types.ManufacturerDetails), memoreaza detaliile de producator in cazul in care contul de utilizator are rolul de Manufactuer

#### Functii pentru interactiunea cu obiecte din contractul inteligent Users

Functia care verifica daca un utilizator salvat in sistem are un anumit rol este functia has. Aceasta primeste doi parametrii, rolul utilizatorului, role (Types.UserRole), si identificatorul contului, account(address). Primul lucru verificat este parametrul account care trebuie sa nu fie nul, mai departe, daca adresa contului este una valida, este accesat obiectul users si se verifica daca utilizatorul cu adresa account are acelasi rol ca parametrul role. In cazul in care rolul utilizatorului este cel precizat in parametru functia returneaza true, in caz contrar va returna false.De aemenea, un alt aspect important este ca functia este o una interna, adica nu poate fi apelata din exteriorul contractului inteligent, este folosita doar de catre alte functii din acelasi contract.

Functia interna care este responsabila de adaugarea unui utilizator in sistem, este add. Aceasta primeste un parametru care specifica detaliile contului de utilizator, user (UserDetails). Initial functia verifica daca adresa din contului este una valida, dupa care se verifica daca exista deja in sistem un cont cu acelasi rol. Acest lucru este realizat cu ajutorul functiei has, care a fost descrisa anterior. Mai departe, in cazul in care totul este in regula, se incrementeaza contorul de monitorizare a numarului de utilizatori, usersCount, dupa care contul este salvat in contractul inteligent prin adaugarea lui user in obiectul users. In final, este emis un eveniment care semnaleaza schimbarea in blockchain, ca un nou utilizator a fost creat cu succes. Evenimentul este NewUser, caruia ii sunt asignate detaliile contului creat.

Functia interna care adauga un utilizator in functie de rolul pe care il detine este \_addUserByRole. Aceasta primeste ca parametru un obiect ce reprezinta un cont de utilizator, user(UserDetails). Functia identifica rolul prin comparatii, iar in functie de tipul acestuia, identificatorul contului, id, este adaugat la una dintre listele manufactuerList, suppliersList, vendorsList sau clientsList. Mai departe contul este adaugat in lista globala de utilizatori prin apelul functiei add. In cazul in care functia nu identifica nici un rol valid, este returnat un mesaj de eroare si totul ramane neschimbat.

Functia folosita pentru adaugarea de utilizatori este \_addUser. Aceasta are doi parametrii, contul de utilizator, user(UserDetails) si identificatorul contului curent. Functia este apelata in contractul intelgent SupplyChain. Aceasta functie verifica parametrii myAccount si user asigurandu-se ca nu contin adrese nule, dupa care se asigura ca rolul utilizatorului cu adresa myAccount este de Admin pentru ca doar administratorul ar trebui sa fie capabil sa introduca utilizatori noi, iar in caz de succes este apelata mai departe functia \_addUserByRole cu parametrul user.

Functia pentru inregistrarea unui cont de utilizator este \_register, aceasta poate fi apelata de un utilizator care nu este inregistrat in sistem. Functia prezinta doi parametrii, detaliile contului de utilizare, user(UserDetails), si adresa contului de Ethereum, myAccount(address), care va fi folosita pe post de identificator. In functie se verifica daca adresa utilizatorului care a apelat functia este acceasi cu cea din user.id, pentru ca dorim ca utilizatorul sa creeze un cont cu propria adresa. Mai departe se introduce noul cont cu ajutorul unui apel catre functia \_addUserByRole cu parametrul user.

Functia de obtinere a detaliilor unui cont de utilizator este get, o functie care primeste ca parametru un identificator al unui cont, account(address), si returneaza un obiect de tipul UserDetails prin accesarea mapping-ului users la adresa account. Inainte de accesarea obiectului users, se verifica daca account este nul si daca utilizatorul cu identificatorul account exista.

Functia care returneaza detaliile unui producator este getManufacturerDetails, aceasta are un singur parametru, identificatorul contului, account(address). Rezultatul este obtinut prin accesarea mapping-ului manufacturerDetailList cu ajutorul identificatorului account, iar valoarea returnata un obiect de tipul ManufacturerDetail.

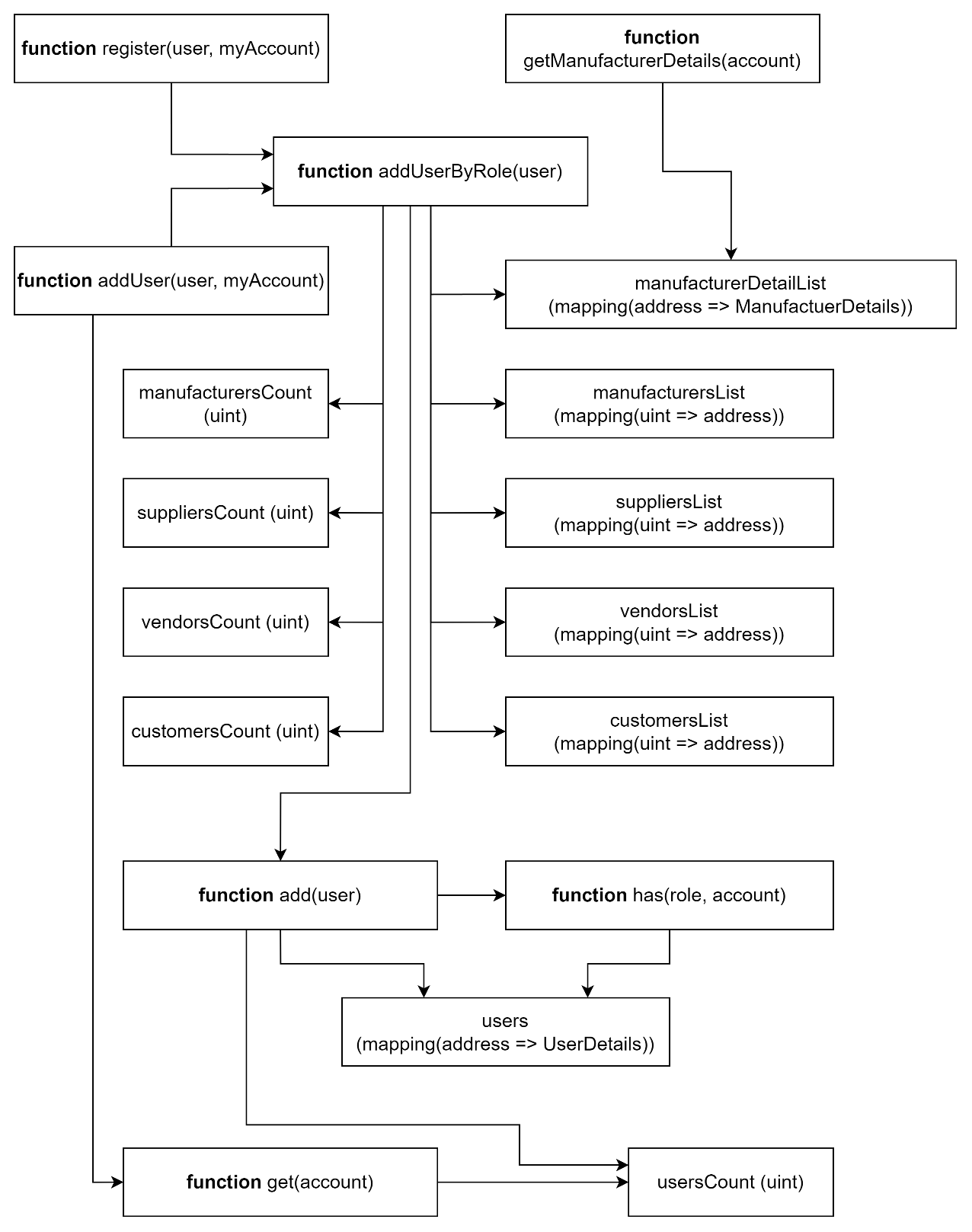


Figura 5.3 Diagrama de interactiune intre functiile si obiectele contractului inteligent Users

### Contractul inteligent destinat administrarii produselor – Products

#### Obiecte utilizate pentru stocarea produselor in sistemul de trasabilitate

Contractul inteligent al produselor, Products, contine obiecte care sunt repsonsabile de memorarea tipurilor de produse, produselor existente in sistem, produsele existente in stocul utilizatorilor si retetelor disponibile pentru utilizare:

* productTypes – memoreaza toate tipurile de produse care pot fi folosite in sistem, este un obiect de tip mapping(uint => Types.ProductType)
* products – memoreaza toate produsele pe baza codurilor de bare unice, este un obiect de tip mapping(string => Types.Product)
* userLinkedStockItems – memoreaza o lista de obiecte StockItem, care sunt produsele din stocul utilizatorului, lista este identificata dupa adresa utilizatorului, este un obiect de tip mapping(address => Types. StockItem[])
* productCounter – numara cate produse au fost create pentru un anumit utilizator, este un obiect de tip mapping(address => uint)), folosit in generarea codurilor de bare
* stockItemCounter – este un contor pentru numarul de produse din stocul unui utilizator, este un obiect de tip mapping(address => uint)
* recipeIngredients – stocheaza lista de ingrediente pentru fiecare reteta in parte, identificatorul retetei ne ofera mapping-ul de ingrediente, iar indexul ingredientului ne ofera ingredientul dorit, este un obiect de tip mapping(uint => mapping(uint => Types.RecipeIngredient))
* recipes – memoreaza fiecare reteta dupa identificatorul acesteia, este un obiect de tip mapping(uint => Types.Recipe)
* recipeCounter – este contorul retetelor, ajuta la indexarea mapping-ului recipes, este un obiect de tip uint
* parentProducts – salveaza o lista de coduri de bare ale unui produs, ajuta in reconstruirea traseului unui produs, este un obiect de tip mapping(string => string[])

#### Obiecte utilizate pentru stocarea trasferurilor din cadrul sistemului de trasabilitate:

* transfers – salveaza toate transferurile create de utilizatori, este un obiect de tip mapping(uint => Types.Transfer)
* transferCount – este contorul transferurilor, ajuta la indexarea transferurilor in mapping-ul transfers, este un obiect de tip uint
* accountTransfers – salveaza pentru fiecare utilizator o lista de identificatori de transferuri ce apartin contului respectiv, este un obiect de tip mapping(address => uint[])
* accountTransferCount – salveaza contorul de transferuri ale unui utilizator, este folosit pentru indexarea listei returneate din mapping-ul accountTransfers, este un obiect de tip mapping(address => uint)

#### Evenimentele emise de contractul inteligent Products in urma unor modificari in sistemul de trasabilitate

Evenimentele din acest contract inteligent sunt folosite pentru a semnala crearea cu succes a tipurilor de produse, a produselor, a retetelor si starea transferului de produse, acestea sunt:

* NewProduct – este emis in momentul in care este creat un produs de baza, acesta contine detalii despre produsul creat: numele produsului (name), numele producatorului (manufacturerName), codul de bare (barcodeId), data de producere (manDateEpoch) si data de expirare (expDateEpoch)
* ComposedProduct – este emis in momentul in care este creat un produs compus, acesta contine informatii despre produs, dar si despre compozitia acestuia. Contine aceleasi informatii ca NewProduct, la care se mai adauga lista de produse parinte (parentProducts) care sunt codurile de bare a produselor folosite in crearea produsului
* NewProductType – este emis cand se creaza un nou tip de produs, acesta contine informatii precum numele (name) si identificatorul tipului de produs (id).
* NewRecipe – este emis atunci cand este creata o noua reteta, aceasta contine informatii despre identificarea retetei (id), rezultatul transformarii bazate pe reteta (resultTypeId) care este o referinta spre productTypes si numele produsului rezultat (resultTypeName)
* ObjectTransferred – este emis in momentul initierii, acceptarii sau refuzului unui transfer, acesta contine date despre id-ul transferului (transferId), expeditorul transferului (sender), destinatarul transferului (receiver), codul de bare al produsului (barcodeId), cantitatea de produs (quantity) si statusul transferului (status)

#### Functii pentru gestionarea produselor din sistemul de trasabilitate

Functia pentru salvarea unei retete este \_addRecipe, care acceseaza mapping-ul recipes la pozitia recipeCounter si salveaza obiectul primit ca parametru, recipe(Types.Recipe), dupa care emite un eveniment de tipul NewRecipe cu informatiile necesare, iar in final incrementeaza contorul care monitorizeaza numarul de retete, recipeCounter.

Functia pentru adaugarea mai multor retete este \_addRecipeList, aceasta primeste ca parametru o lista de retete, recipeList(Types.Recipe[]), care este parcursa si pentru fiecare element este apelata functia \_addRecipe.

Functia responsabila pentru a crea un nou produs este \_addProductType. Aceasta creaza un obiect de tipul ProductType cu ajutorul parametrului de intrare productType(Types.ProductTypeAddDTO) si a contorului productTypeCounter, dupa care salveaza obiectul in productTypes la pozitia oferita de productTypeCounter. Mai departe se emite evenimentul NewProductType si se incrementeaza contorul.

Pentru a adauga mai multe tipuri de produse se poate utiliza functia \_addProductTypeList care primeste ca parametru o lista de tipuri de produse, productTypeList (Types.ProductTypeAddDTO). Se parcurge lista de si se apeleaza functia descrisa anterior \_addProductType cu fiecare element din lista.

Adaugarea unui produs de baza se realizeaza cu ajutorul functiei \_addProduct. Aceasta are ca parametrii un obiect ce reprezinta produsul, productAddDto (Types.ProductAddDTO), un obiect care contine detalii despre producator, manufacturerDetails(Types.ManufacturerDetails), numele producatorului, manufacturerName(string) si adresa contului care a apelat functia, myAccount(address). Initial functia verifica daca identificatorul producatorului care a creat produsul coincide cu identificatorul contului care a apelat functia.

Urmatorul pas este generarea codului de bare folosind manufacuterDetails si numarul de produse create pana in acest moment de utilizatorul cu adresa myAccount dat de productCounter. Codul de bare returnat de apelul functiei generateBarcode este salvat in variabila barcodeId.

Mai departe se creaza un nou obiect de tipul Product, acesta este construit pe baza parametrilor de intrare productAddDto, manufacturerName, myAccount si a codului de bare generat anterior. Dupa creare obiectului, acesta este salvat in mapping-ul global de produse, products, utilizand cheia de identificare barcodeId.

Asocierea cu contul utilizatorului care a adaugat noul produs incepe prin incrementarea contorului productCounter si crearea unui obiect de tipul StockItem cu ajutorul codului de bare, barcodeId si a cantitatii de produs adaugat, productAddDto.batchCount. In continuare se adauga obiectul stockItem in lista din mapping-ul userLinkedStockItems, dupa care totul se finalizeaza cu incrementarea contorului produselor din stoc, stockItemCounter. Inainte de a iesi din functie, se emite un event de tipl NewProduct in care se introduc datele produsului nou creat.

Crearea unui produs pe baza unei retete existente in sistem se realizeaza cu functia \_createProduct. Aceasta primeste patru parametrii, identificatorul retetei, recipeId(uint), contul de utilizator, user(Types.UserDeails) si detaliile producatorului, manufacturerDetails (Tyes.ManufacturerDetails).

Primul lucru care se realizeaza este preluarea retetei din lista de retete cu ajutorul identificatorului recipeId, urmand ca apoi sa se intializeze o lista de string-uri, \_parentProducts, in care vor fi salvate codurile de bare ale ingredientor. De asemenea, se initializeaza si un contor, ingredientsCount, pentru a verifica daca in stocul utilizatorului exista toate ingredientele din reteta folosita.

Algoritmul pentru aceasta functionalitate a fost descris anterior la punctul 4.2. Implementarea algoritmului se rezuma la verificarea ingredientelor folosind mapping-urilor reciepIngredients si userLinkedStockItems , descresterea cantitatii produselor din stocul utilizatorului din userLinkedStockItem, salvarea codurilor de bare ale ingredientelor in \_parentProducts si incrementarea contorului ingredientsCount.

Dupa parcurgeri se verifica contorul ingredientsCount, iar daca totul este in regula se creaza un obiect de tipul Product, product, care primeste valori din recipe, user si rezultatul apelului generateBarcode care genereaza codul de bare, dupa care produsul este salvat in mapping-ul products.

Mai departe se salveaza lista \_parentProducts in mapping-ul parentProducts, identificat de prin codul de bare al produsului creat, se salveaza un obiect de tipul StockItem in lista din mapping-ul userLinkedStockItems si se incrementeaza contorul stockItemCounter. In final se emite un event de tipul ComposedProduct in care sunt transmise numele produsului, numele producatorului, codul de bare, data de productie, data de expirare si lista de produse parinti.

Toate functiile legate de adaugarea retetelor, tipurilor de produse, produselor si crearea produselor pe baza unei retete interactioneaza cu obiectele de tip mapping si uint existente in contractul inteligent, diagrama de interactiune dintre acestea poate fi vizualizata in Figura 5.4.

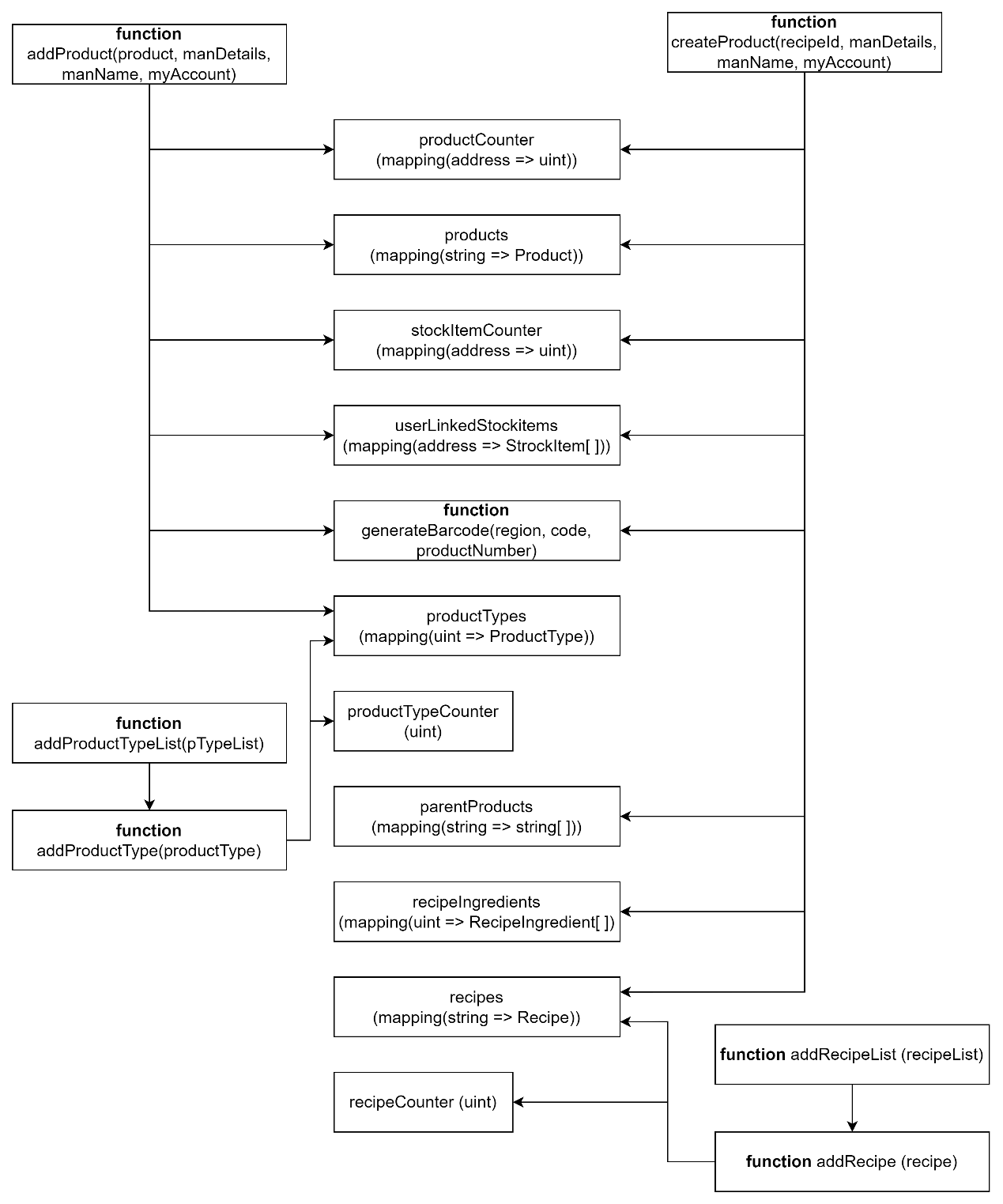


Figura 5.4 Diagrama de interactiune a functiilor si obiectelor folosite pentru adaugarea sau crearea de produse

#### Functii pentru transferul produselor din sistemul de trasabilitate

Functiile responsabile pentru initierea, acceptarea si refuzul unui transfer a unui produs de la un utilizator catre altul pot fi vizualizate in Figura 5.5.

Initierea unui transfer se realizeaza in functia requestTransfer, aceasta primeste ca parametrii codul de bare al produsului, barcodeId(string), cantitatea produsului, quantity(uint) si identificatorul destinatrului, receiver(address). In aceasta functie se creaza un nou obiect de tip Transfer, newTransfer, care ii sunt atribuite identificatorul numarul de transferuri (transferCount), adresa expeditorului(msg.sender), adresa destinatarului(receiver), codul de bare al produsului (barcodeId), cantitatea produsului (quantity) si statusul transferului, care pentru ca abia se creaza este unul de asteptare (ObjectStatus.Pending).

Dupa crearea obiecului newTransfer, acesta este salvat in mapping-ul de transferuri transfers la pozitia transferCount. Mai departe se asociaza transferul creat cu contul destinatarului, prin salvarea identificatorului de transfer in mapping-ul accuntTransfers. In final se incrementeaza contoarele accountTrnasferCount si transferCount, dupa care se emite un event de tipul ObjectTransferred.

Functia folosita de catre destinatar pentru a accepta un transfer care este in asteptare este acceptTransfer. Aceasta verifica daca transferul identificat prin parametrul transferId(uint) are ca destinatar adresa utilizatorului care a initiat apelul functiei. Se mai verifica daca transferul este unul in asteptare, pentru a nu permite unui utilizator sa accepte transferuri daca acestea au fost deja acceptate.

Mai departe se apeleaza functia care va realiza modificarea stocurilor utilizatorilor intre care se realizeaza transferul, transferOwnership. Daca totul este in regula, se modifica statusul transferului in ObjectStatus.Accepted, iar in final se emite un event de tipul ObjectTransferred.

Functia opusa celei de acceptare, refuseTransfer, permite unui destinatar sa refuze un transfer. In aceasta functie se fac aceleasi verificari pentru identificarea utilizatorului si a transferului, dupa care se schimba statusul obiectului din mapping-ul transfers, identificat de parametrul functiei, transferId(uint), in ObjectStatus.Refused.

Functia care permite unui utilizator sa investigheze statusul unui transfer este getTransferStatus, aceasta acceseaza mapping-ul transfers la pozitia transferId(uint) si returneaza statusul accesand parametrul status al obiectului de tip Transfer accesat.

Transferul de produse se realizeaza in functia transferOwnership(address sellerId, address buyerId, string barcodeId, uint quantity). In primul rand se acceseaza produsele din stocul expeditorului dupa adresa acestuia, sellerId. Mai departe se identifica pozitia din stoc a produsului cu codul de bare barcodeId si se verifica daca cantitatea ce se doreste sa fie transferata este disponibila in stoc. Daca verificarile esueaza, functia anuleaza efectele de pana acum, iar in caz de succes se incepe modificarea cantitatilor din stocurile utilizatorilor.

Prima data se scade cantitatea transferata, quantity, din cantitatea din stocul expeditorului prin accesarea stocului cu ajutorul mapping-ului userLinkedStockItems. In continuare, se initializeaza variabila buyerAlreadyHasStockItem cu false, dupa care se parcurge stocul destinatarului si se compara produsele existente cu codul de bare al produsului primit, barcodeId, daca produsul exista deja in stocul utilizatorului, atunci se creste cantitatea produsului din stoc cu valoarea quantity, iar variabila buyerAlreadyHasStockItem este setata pe true.

Daca utilizatorul nu are in stoc produsul ce se transfera, adica buyerAlreadyHasStockItem este false, atunci se creaza un nou obiect de tipul StockItem, transferredItem, care primeste ca proprietati codul de bare, barcodeId si cantitatea de produs, quantity. Dupa crearea obiectului, acesta se salveaza in mapping-ul userLinkedStockItems la adresa destinatarului, buyerId, si se incrementeaza numarul de produse din stoc.

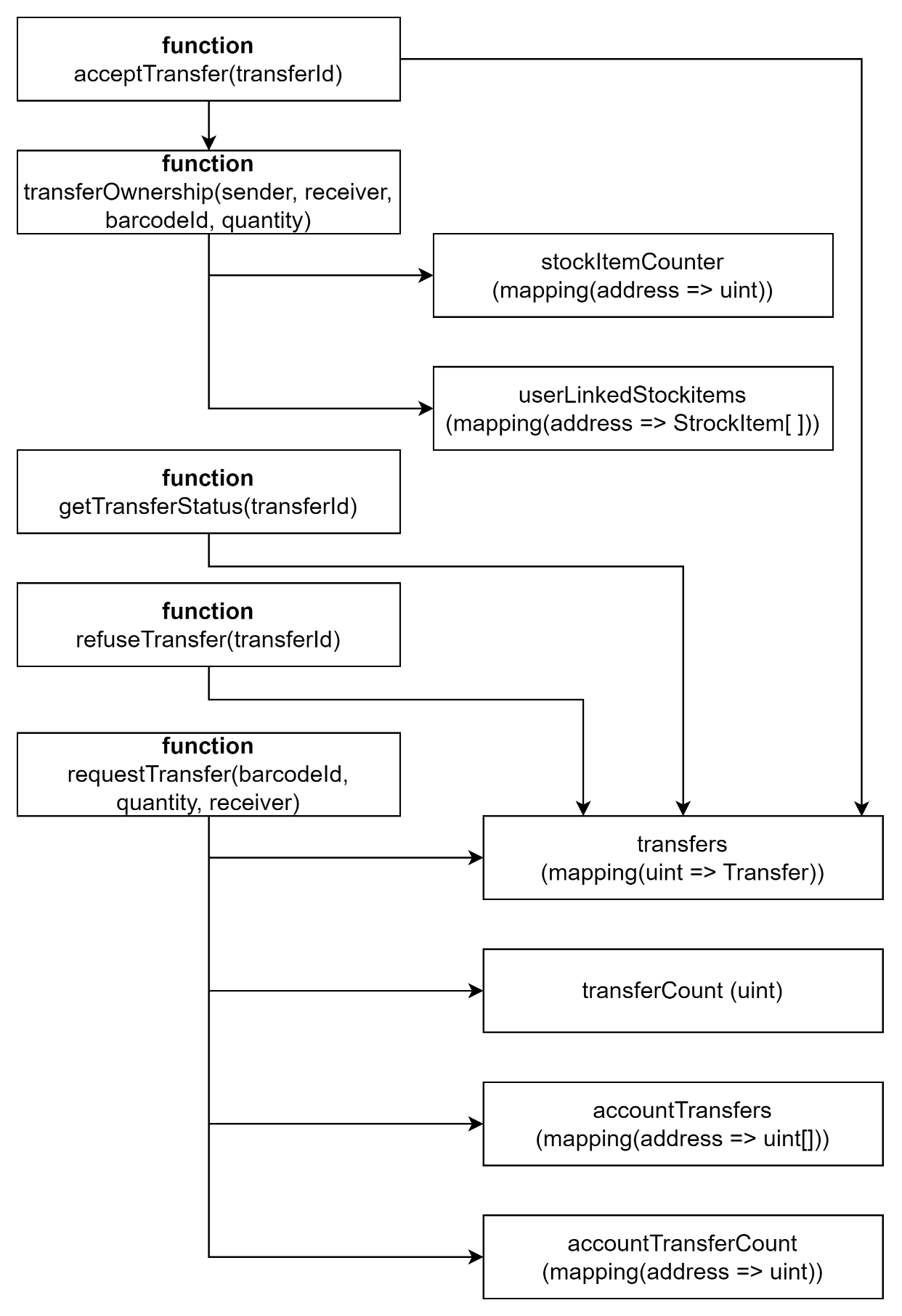


Figura 5.5 Diagrama de interactiune a obiectelor si functiilor utilizate pentru transferul de produse intre utilizatori

## Front-end: Interfata aplicatiei de trasabilitate

### Clase folosite pentru definirea de obiecte utilizate in comunicarea cu contractele inteligente – modulul models

In acest modul sunt fisiere in care sunt definitii ale claselor ce definesc modelele de date, acestea sunt folosite pentru o comunicare usoara intre componentele proiectului.

Clasa Product contine parametrii care sunt intalniti si in structura de date Product din biblioteca Types din partea de back-end a proiectului, parametrii clasei sunt: name, productTypeId, barcodeId, manufacturerName, manufacturerId, manufacturingDate, expirationDate, dar contine si parametrul optional quantity care este folosit pentru numarul de produse din stocul utilizatorului. Constructorul clasei are doi parametrii, product si quantity.

Clasa ProductType contine parametrii: id, name si details si contine un constructor care atribuie valori parametrilor prin extragerea datelor din obiectul productType primit ca intrare in constructor.

Clasa Recipe are parametrii: id, ingredients, ingredientsCount, quantityResult, resultTypeId si resultTypeName. In constructor extrage date din parametrul de intrare recipe, folosind operatorul [ ] si converteste unele date la number pentru a evita erori de tipuri de date.

Clasa RecipeIngredient are parametrii: productQuantity, productTypeId, recipeId si productType, primii trei parametrii sunt initializati in constructor prin extragerea datelor din variabila ingredient, iar parametrului produtType ii este atribuit un obiect de tipul ProductType.

In UserRoles.js este clasa UserRoles care contine atribute statice care definesc rolurile posibile pentru un actor al sistemului, Admin, Manufacturer, Supplier, Vendor, Customer. Acestea au valori de la 0 la 4 si sunt o echivalare a enumerarii UserRoles definita in partea de back-end a proiectului in biblioteca Types. In acest fisier sunt definite doua functii de mapare, userFromString care mapeaza un string la o valoarile din clasa UserRoles, si userRoleToString care mapeaza o valoare din clasa UserRoles la un string.

### Clase utilizate pentru apelarea functiilor din contractele inteligente – modulul contracts

Avand in vedere ca sunt trei contracte inteligente, aici sunt definite trei clase care comunica cu acestea, o clasa pentru fiecare contract: ProductsContract, SupplyChainContract, UsersContract.

Constructorii claselor sunt asemantori, in fiecare este initializat parametrul \_provider. Pentru a realiza conexiunea cu contractul inteligent, se salveaza intr-un obiect detaliile contractului inteligent, acestuia ii sunt atribuite adresa contractului, abi-ul contractului si un obiect de tip semnatar, care poate fi accesat din parametrul \_provider cu ajutorul metodei getSigner.

Dupa ce conexiunea cu contractul inteligent a fost stabilita, se pot apela functiile din contractul ineligent prin utilizarea obiectului de tip Contract (usersContract, supplyChainContract sau productsContract).

In clasa SupplyChainContract se pot identifica functii care apeleaza contractul inteligent SupplyChain. Majoritatea functiilor din clasa primesc parametrii si apeleaza o functie din contractul inteligent, asteapta raspunsul acesteia, in cazul unei erori o semnaleaza, iar in caz de succes verifica rezultatul cu ajutorul functiei handleTransaction(transaction). Aceasta functie ajuta la verificarea statusului operatiunii facute si ajuta la mecanismul de reimprospatare a datelor.

#### Interactiunea cu contractul inteligent Users

Clasa UsersContract, apeleaza functiile din contractul inteligent Users.

Functia getCurrentUser face un apel pentru a identifica conturile de Ethereum disponibile, iar prin accesarea elemetului de pe pozitia zero va fi returnat contul curent. Mai departe se returneaza apelul functiei getUser cu parametrul adresa contului curent.

Functia getUser, primeste un parametru, address, si il utilizeaza pentru a apela functia get din contractul inteligent Users si returneaza obiectul rezultat.

Functia getUserList determina o lista a utilizatorilor, mai putin cea a contului curent. Aceasta incepe prin determinarea contul curent cu ajutorul functiei getCurrentUser descrisa anterior, dupa care face un apel catre userCount pentru a determina numarul de utilizatori din sistem si rezultatul se salveaza in variabila count. Mai departe se initializeaza o lista vida, userList, se intra intr-o bucla de la 0 la count, in care este apelata functia usersByIndex din contractul inteligent pentru a obtine adresa unui utilizator in functie de index. Rezultatul este folosit in determinarea utilizatorului care se face cu functia getUser, acest ultim rezultat este adaugat in userList. In final se returneaza lista de utilizatori.

Pentru a determina numarul de utilizatori din sistem se poate apela functia getUsersCount care face un apel catre functia din contractul inteligent, usersCount.

Functia register inregistreaza un nou cont de utilizator transmis prin parametrul user. Functia determina contul curent, dupa care apeleaza functia register din contractul inteligent.

#### Interactiunea cu contractul inteligent Products

Clasa ProductsContract are functii care preia toate tipurile de evenimentele emise de catre contractul inteligent Products. Pentru a face acest lucru se foloseste functia queryFilter din obiectul productsContract, care primeste ca parametru numele eventului dori. Functia queryFilter este apelata cu parametrii NewProduct, NewProductType, NewRecipe, ComposedProduct, si ObjectTransferred in functiile getProductsEvents, getProductTypeEvents, getRecipeEvents, getComposedProductEvents si getObjectTransferred. Rezultatul functiei queryFilter este parsat si trimis mai departe pentru a putea fi folosit in componentele responsabile de afisarea rezultatelor.

Alte functii intalite in aceasta clasa apeleaza direct functii din contractul inteligent Products si transmit mai departe rezultatul acestora. Aceste functii sunt: getRecipeCounter(), getProduct(barcodeId), requestTransfer(barcodeId, quantity, receiver), acceptTransfer(transferId), refuseTransfer(transferId), getTransferStatus (transferId).

Functia getRecipeList returneaza o lista cu toate retetele disponibile in contractul inteligent. In primul rand, initializeaza o lista vida, recipes, determina cate retete exista apeland functia recipesCounter si salveaza valoarea in variabila recipeCounter. Mai departe se intra intr-o bucla, in care se extrag pe rand retetele cu ajutorul functiei recipes(index), iar pentru fiecare reteta se extrag ingredientele retetei cu ajutorul altei bucle, in care se apeleaza functiile recipeIngredients(recipeIndex, ingredientIndex) si productTypes(productTypeId) si se salveaza rezultatul acestora in lista de ingrediente. Dupa iesirea din a doua bucla se adauga reteta in lista recipes, iar dupa iesirea din prima bucla se returneaza lista respectiva.

Functia getProductType() returneaza o lista de tipuri de produse construind o lista apeland repetitiv functia productTypes(index) din contractul inteligent si salvand rezultatul in obiecte de tip ProductType.

Functa getProductlist(userId) determina numarul de produse din stocul utilizatorului, dupa care construieste o lista de produse prin apeluri repetitive catre functiile userLinkedStockItems(userId, itemIndex) salvand rezultatele acestora in obiecte de tip Product care in final sunt adaugate la lista productList.

Functiile trackProduct(barcodeId) si parentProduct(barcodeId) sunt o implementare a algoritmului de refacere a traseului unui produs, explicat anterior la punctul 4.4. Functia trackProduct apeleaza functia getProducts(barcodeId) si functia parentProducts(barcodeId) din clasa. Functia parentProducts(barcodeId) utilizeaza functiile products(barcodeId) si parentProducts(barcodeId, ingredientIndex) din contractul inteligent pentru a determina produse si lista de produse a unui produs. La fel cum este prezentat si in algoritm functia intra intr-un apel recursiv, iar in final produsele sunt salvate in lista parentProductList.

Accesarea transferurilor unui utilizator este realizata in functia accountTransfers, se identifica numarul de transferuri prin apelul accountTransferCount din contractul inteligent Products, apoi se apeleaza repetitiv functia transfers din Products pentru a a crea lista de transferuri.

### Elemente de prezentare din interfata utilizator

#### Paginile aplicatiei de trasabilitate – modulul pages

Paginile aplicatiei faciliteaza afisarea datelor primite din contractele inteligente, permit utilizatorului sa introduca date si apeleaza functii din contractele inteligente prin intermediul claselor SupplyChainContract, UsersContract si ProductsContract. Toate paginile contin functia loadBlockChainData care este apelata in momentul in care pagina este incarcata pentru prima data. Aceasta functie este folosita pentru a incarca date din contractele inteligente.

Pagina de identificare a unui produs cu ajutorul codului de bare este TrackingPage. Aceasta pagina contine un obiect de tip ProductsContract, \_productsContract utilizat pentru a apela functia de reconstruire traseului unui produs, si un obiect productHistory care va contine istoricul produsului cautat. Pagina afiseaza o componenta care permite introducerea codului de bare si o componenta care afiseaza rezultatul cautarii.

Pagina de profil a unui utilizator, ProfilePage, afiseaza detaliile contului conectat la aplicatia blockchain. Pagina contine un obiect care salveaza detaliile contului, user, si un obiect care realizeaza conexiunea cu contractul inteliget User, \_userContract. Pentru obtinerea datelor utilizatorului se apeleaza functia getCurrentUser, iar datele obtinute sunt transmise catre componenta UserDetails pentru a fi afisate.

Pagina de afisare a retetelor disponibile in sistem este RecipePage, aceasta contine un obiect in care se va salva lista de retete, recipes, in momentul in care se apleaza functia getRecipeList din obiectul \_productContract. Pagina afiseaza doua componente, un text si o matrice de celule care afiseaza datele retetelor.

Pagina de afisare a tipurilor de produse disponibile in sistem este ProductsPage, aceasta contine un obiect in care se va salva lista de tipuri de produse, productTypes. Pagina afiseaza doua componente, un tabel care afiseaza datele din lista productTypes, si o componenta care permite adaugarea de tipuri de produse in sistemul de trasabilitate. Componenta este vizibila doar pentru utilizatorii de tip Admin pentru ca functionalitatea de a adauga tipuri de produse nu este permisa pentru alte tipuri de utilizatori.

Pagina de cautare a unui cont din sistemul de trasabilitate este CompaniesPage. Aceasta contine o componenta care permite introducerea unui identificator, care mai departe va fi utilizat pentru cautarea in lista de conturi din contractul inteligent Users. Datele obtinute in urma acestui apel vor fi afisate in alta componenta.

Conectarea unui actor in sistem se face prin intermediul paginii ConnectPage. In aceasta pagina sunt folosite componente pentru introducerea datelor unui utilizator. Datele completate pot fi transmise mai departe functiei register din clasa UsersContract.

Evenimentele emise de contractele inteligente din blockchain sunt afisate in pagina DashboardPage. Pagina contine instante ale claselor ProductsContract si UsersContract, \_productsContract si \_usersContract. In pagina se apeleaza functia getUserCount din \_userContract, si functiile getProductTypeEvents, getProductEvents si getComposedProductEvent din \_productsContract. Datele obtinute sunt afisate in de text sau sub forma tabelara.

Pagina pentru vizualizarea transferurilor este SellProductPage. Aceasta pagina obtine detaliile contului curent si lista transferurilor asociate contului. Pentru a accepta sau refuza un transfer se pot apela functii din clasa ProductsContract. Datele obtinute sunt afisate sub forma tabelara folosind o componenta. Pe de alta parte, pagina mai contine componente care permit introducerea datelor pentru initierea transferului unui produs. Utilizatorul poate introduce codul de bare al produsului pe care doreste sa il transfere, cantitatea produsului si poate selecta dintr-o lista de utilizatori cui doreste sa ii trimita produsul.

Stocul unui utilizator poate fi vizualizat in pagina MyStockPage. Datele produselor din stoc sunt preluate prin accesarea contractului inteligent si transmise unei componente care afiseaza produsele intr-un tabel. Pe aceeasi pagina se pot adauga produse de baza prin selectarea tipului de produs, a datelor de productie si expirare si a cantitatii sau se pot adauga produse compuse prin selectarea unei retete. Pentru a realiza inregistrarea produselor in sistemul de trasabilitate se folosesc functiile din clasa ProductsContract.

#### Componentele utilizate in aplicatiei de trasabilitate – modulul components

Componentele utilizate in paginile prezentate anterior sunt folosite pentru afisarea datelor sau pentru introducerea datelor de catre utilizatori. Componentele primesc parametrii care contin obiecte sau functii care se apeleaza la apasarea unui buton, aceste sunt impartite in urmatoarele categorii:

* Formulare pentru introducerea datelor si transmiterea acestora prin intermediul unei functi: AddProductForm, AddProductTypeForm, AddUserForm, CreateProductForm, FindUserForm, ProductTrackForm
* Tabele: TransferTable, ProductTable, ProductTypeTable, RecipeTable
* Carduri: ProductCard, RecipeCard
* ProductsGrid
* UserDetails
* SimplePopup
* TransactionSnackbar

# Testare și validare

Teste – unit-testele implementate in aplicatie

Utilizarea propriu-zisa a aplicatiei, testarea functionalitatilor principale

Calcularea pretului platit pentru tranzactii, gas-ul utilizat

Erorirle returnate

# Manual de instalare si utilizare

## Instalarea mediilor de dezvoltare

Atat aplicatia back-end, cat si cea front-end folosesc mediul de dezvoltare Node.js. Acesta este utilizat in versiunea 16.20.2, impreuna cu npm versiunea 8.19.4.

Descarcare Node.js cu versiunea proiectului se poate face de la adresa: https://nodejs.org/en/blog/release/v16.20.2.

Dupa descarcare si instalare, se verifica versiunea de Node.js si de npm instalate folosind in terminal urmatoarele comenzi:

* node -v
* npm – v



Figura 7.1 Verificare instalare Node.js si npm

Mai departe, folosind un terminal deschis in folderul proiectului de back-end se va instala mediul de dezvoltare pentru software-ul Ethereum, Hardhat. Comenzile necesare instalarii si configurarii sunt:

* + npm install --save-dev "hardhat@^2.19.1" "nomicfoundation/hardhat-toolbox@^4.0.0"
  + npm install dotenv --save
  + npm install

In continuare, pentru proiectul de front-end, folosind un terminal deschis in folderul acestuia, se instaleaza toate pachetele si dependintele necesare folosind comanda:

* + npm install

## Instalarea si rularea proiectului de back-end

In folderul proiectului de back-end, pentru a compila contractele inteligente se foloseste urmatoarea comanda:

* + npx hardhat compile

Rezultatele comenzii, contractele inteligente compilate, se gasesc in folderul artifacts/contracts.

Pentru a crea o retea blockchain locala se foloseste urmatoarea comanda:

* npx hardhat node

De asemenea, comanda va genera o lista de zece conturi care se pot folosii pentru a interactiona cu reteaua blockchain. Conturile vor fi folosite mai departe pentru a comunica cu contractele inteligente.

Mai mult, interactiunile cu reteaua blockchain poate fi vizualizate in acest terminal, ceea ce inseamna ca acesta trebuie lasat pornit pentru functionarea proiectului.

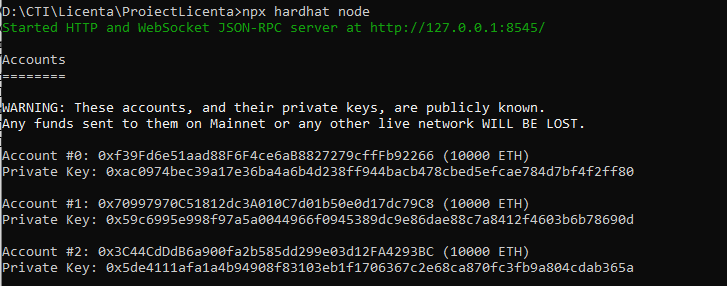


Figura 7.2 Rularea unei retele locale de blockchain

Pasul urmator este incarcarea contractelor inteligente in reteaua locala blockchain, deschisa la pasul anterior. Folosind un nou terminal, se va rula comanda:

* npx hardhat run scripts/deploy\_supply\_chain.js --network localhost

Rezultatul comenzii va afisa adresele contractelor inteligete din reteaua blockchain:

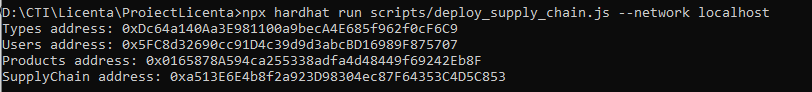


Figura 7.3 Incarcarea contractelor inteligente in reteaua locala de blockchain

Cu acest pas se incheie pornirea proiectului de back-end, datele oferite de comenzile rulate vor fi utilizate in aplicatia front-end.

## Instalarea si rularea proiectului de front-end

Pentru a configura conexiunea cu contractele inteligente, este necesar ca aplicatia sa stie adreasa acestora in reteaua blockchain. Adresele pot fi preluate de la ultimul pas pentru instalarea proiectului de back-end ilustrate in Figura 7.3. Acestea trebuie introduse in fisierul src/config.js dupa cum se poate observa in Figura 7.4.

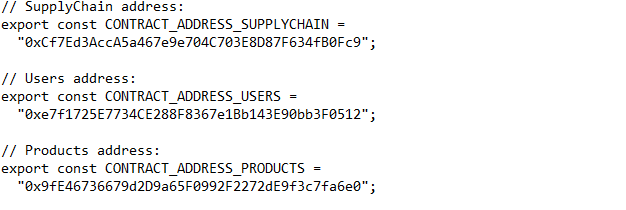


Figura 7.4 Fisierul de configurare al contractelor inteligente src/config.js

Mai departe proiectul poate fi rulat folosind comanda:

* npm start

Comanda ruleaza aplicatia web la adresa http://localhost:3000/, pe care o si deschide intr-un browser web.

## Instalarea si configurarea extensiei MetaMask

Pentru a interactiona cu contractele inteligente prin intermediul aplicatiei web, este nevoie de instalarea extensiei MetaMask in browser-ul folosit. Aceasta poate si gasite magazinul web de extensii ale browser-ului, spre exemplu in Microsoft Edge extensia poate fi gasita la adresa:

* https://microsoftedge.microsoft.com/addons/search/MetaMask

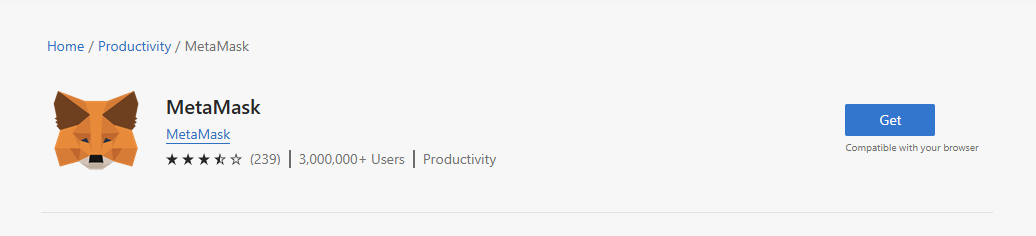


Figura 7.5 Pagina de instalare a extensiei MetaMask

Dupa instalarea extensiei, este nevoie de crearea unui cont folosind pasii din Figura 7.6.

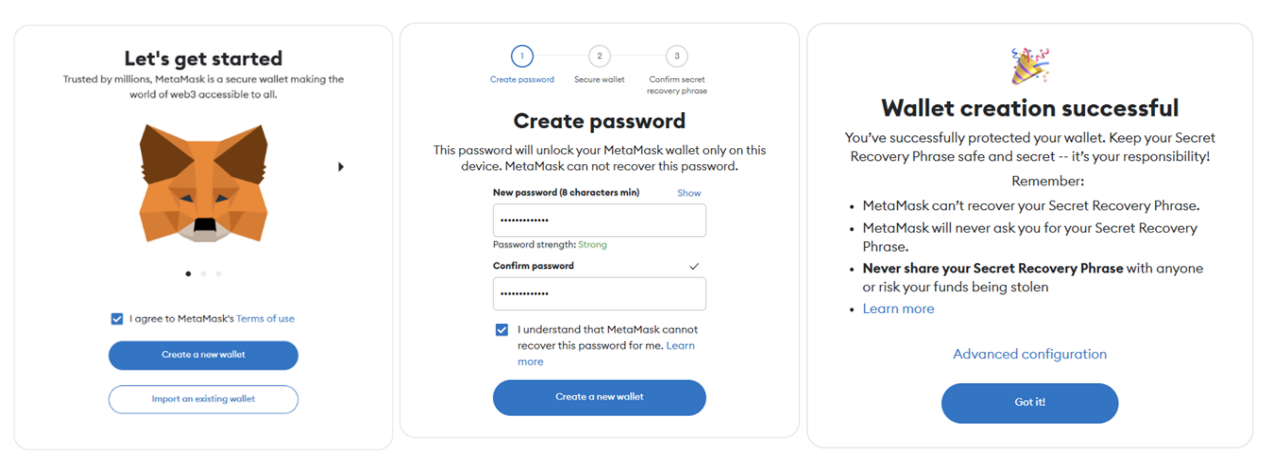


Figura 7.6 Pasii de crearea a unui portofel digital in MetaMask

Mai departe, reteaua locala blockchain trebuie inregistrata manual in MetaMask. Se deschide meniul extensiei, precum se face in Figura 7.7, dupa care se apasa pe iconita punctului 0, dupa care se apasa pe butonul de la punctul 3 pentru a putea accesa meniul de setari. Mai departe se selecteaza optiunea pentru retele, adica punctul 4, iar pentru a putea adauga o retea noua se apasa pe butonul din punctul 5.

In pagina deschisa cum este cea din Figura 7.8 se selecteaza adaugarea manual a unei retele prin apasarea butonului de la punctul 6. In continuare se vor introduce datele retelei locale blockchain:

* Network name: Hardhat Localhost
* New RPC URL: http://127.0.0.1:8545/
* Chain ID: 31337
* Currency symbol: GO

Dupa introducerea datelor se salveaza reteaua folosind butonul ilustrat de punctul 7 din Figura 7.8.

Avand reteaua setata, utilizarea acesteia se poate face prin apasarea butonului de la pasul 1 din Figura 7.7, dupa care se selecteaza reteaua apasand pe aceasta, cum este indicat de numarul 8 din Figura 7.8

Ultimul pas este cel de adaugare a unui cont din cele generate in momentul initierii retelei locale. Se apasa pe butonul indicat de punctul 2 ilustrat in Figura 7.7, dupa care se urmeaza punctele 9 si 10 din Figura 7.9. In continuare se introduce o cheie privata a unui cont, acestea pot fi gasita in terminalul initierii retelei blockchain, cum se poate observa in Figura 7.2. Dupa introducerea cheii, se apasa pe butonul ilustrat de punctul 11 din Figura 7.9, iar selectarea contului introdus poate fi facuta folosind punctul 2 din Figura 7.7, urmat de punctul 12 din Figura 7.9.

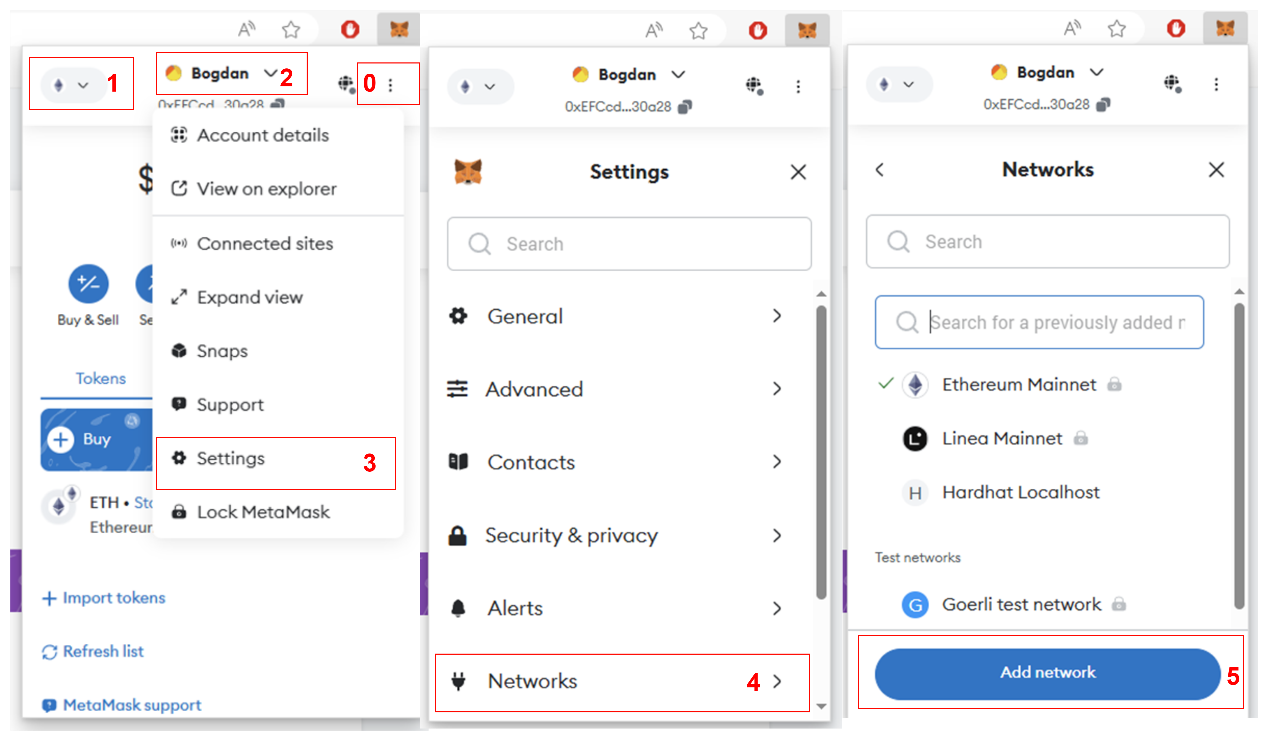


Figura 7.7 Pasii pentru adaugarea unei retele blockchain in MetaMask (1)

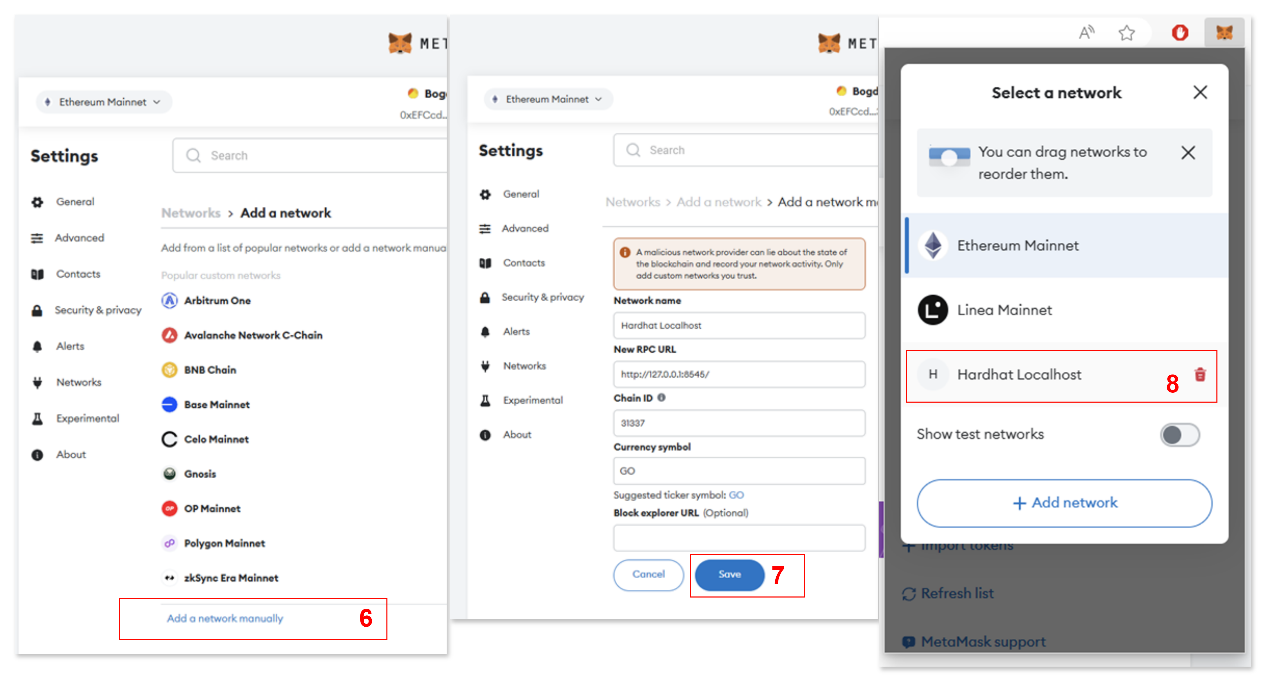


Figura 7.8 Pasii pentru adaugarea unei retele blockchain in MetaMask (2)

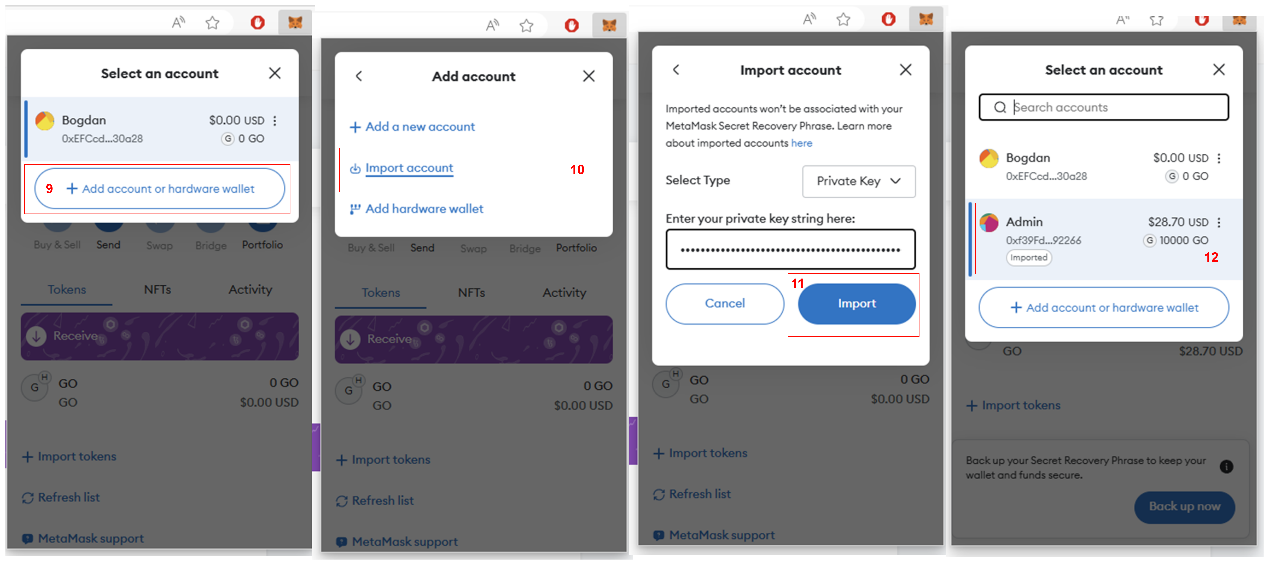


Figura 7.9 Pasii pentru adaugarea unui cont in MetaMask

## Utilizarea sistemului de trasabilitate

Pentru utilizarea sistemului, acesta trebuie instalat folosind instructiunile oferite in pasii anteriori. Utilizarea sistemului se face cu ajutorul aplicatiei de interfata care poate fi accesata in browser la adresa http://localhost:3000/.

### Conectarea unui actor in sistemul de trasabilitate

### Vizualizarea stocului si inregistrarea unui produs

De baza si compus

### Transferul unui produs

Initiere transfer

Acceptare Transfer

### Vizualizarea istoricului unui produs

Cautare pe baza codului de bare

# Concluzii

# Bibliografie

|  |  |
| --- | --- |
| [1] | A. Singh, A. Gutub, A. Nayya și M. K. Khan, „Redefining food safety traceability system through,” *Multimedia Tools and Applications,* p. 21243–21277, 2023. |
| [2] | H. Treiblmaier și M. Garaus, „Using blockchain to signal quality in the food supply chain: The impact on consumer purchase intentions and the moderating effect of brand familiarity,” *International Journal of Information Management,* vol. 68, p. 102514, 2023. |
| [3] | Q. Tao, Z. Cai și X. Cui, „A technological quality control system for rice supply chain,” *Food and Energy Security,* vol. 12, 2023. |
| [4] | UNDP, „Blockchain for Agri-Food Traceability,” *Singapore: UNDP Global Centre for Technology, Innovation and Sustainable Development,* 2021. |
| [5] | R. Gravani, S. Bratager, A. Kennedy, J. McEntire, J. McGill, A. Sayler și F. Yiannas, „Food Traceability: Current Status and Future Opportunities,” vol. 71, 2023. |
| [6] | F. Anastasiadis, I. Apostolidou și A. Michailidis, „Food Traceability: A Consumer-Centric Supply Chain Approach on Sustainable Tomato,” *Foods,* vol. 10, p. 543, 2021. |
| [7] | S. Joshi, M. Sharma, Y. E. B., Y. Kazancoglu, S. Luthra și M. Prasad, „Assessing Supply Chain Innovations for Building Resilient Food Supply Chains: An Emerging Economy Perspective,” *Sustainability,* vol. 15, p. 4924, 2023. |
| [8] | G. Jianli, C. Korhan și T. Ravi, „An IOT and Blockchain Approach for Food Traceability System in Agriculture,” *Scalable Computing: Practice and Experience,* vol. 22, pp. 127-137, 2021. |
| [9] | M. v. Hilten, G. Ongena și P. Ravesteijn, „Blockchain for Organic Food Traceability: Case Studies on Drivers and Challenges,” *Frontiers in Blockchain,* vol. 3, 2020. |
| [10] | C. Zhang, Y. Gong și S. Brown, Blockchain Applications in Food Supply Chain Management, Cham, Switzerland: Springer Nature Switzerland AG, 2023. |
| [11] | A. Alamsyah, S. Widiyanesti și P. Wulansari, „Blockchain traceability model in the coffee industry,” *Journal of Open Innovation: Technology, Market, and Complexity,* vol. 9, 2023. |
| [12] | P. Simon, M. David și L. Georgios, „Are Distributed Ledger Technologies the panacea for food traceability?,” *Global Food Security,* vol. 20, pp. 145-149, 2019. |
| [13] | G. Cachon și M. Lariviere, „Supply chain coordination with revenue sharing contracts: strengths and limitations,” *Management Science,* vol. 51, pp. 30-44, 2005. |
| [14] | G. Boella și L. van der Torre, „Contracts as Legal Institutions in Organizations of Autonomous Agents,” în *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'04)*, New, 2004. |
| [15] | G. Boella, J. Hulstijn și L. van der Torre, „A Synthesis Between Mental Attitudes and Social Commitments in Agent Communication Languages,” în *Intelligent Agent Technology 05 (IAT 2005)*, Compiegne, 2005. |
| [16] | Software Freedom Conservancy, „The Selenium Browser Automation Project,” [Interactiv]. Available: https://www.selenium.dev/. [Accesat 17 martie 2021]. |
| [17] | C. P. Pfleeger, S. L. Pfleeger și J. Margulies, Security in Computing, 5th Edition, Pearson, 2015. |

# Anexa 1

# Anexa 2

# Anexa 3