



Universitatea
Transilvania
din Brașov
FACULTATEA DE MATEMATICĂ
ȘI INFORMATICĂ

LUCRARE DE LICENȚĂ

Platformă pentru simularea de tranzacționări cu jetoane nefungibile

Absolvent: Constantinescu David

Coordonator: Lect. Univ. Dr. Sangeorzan Livia

Brașov, iulie 2022

Cuprins

1 Sectiune introductivă	3
1.1 Tehnologia blockchain	4
1.1.1 Arhitectura unui sistem blockchain	6
1.2 Cryptomonede	7
1.2.1 Moneda romanească Elrond	9
1.3 Jetoane nefungibile	11
1.4 Internetul	14
1.4.1 World Wide Web	15
1.4.2 Aplicațiile Web	16
1.5 Scopul proiectului	16
1.5.1 Obiectivele proiectului	17
2 Tehnologiile folosite	18
2.1 Java	18
2.2 HTML	19
2.2.1 Etichetele HTML	20
2.2.2 Elementele HTML	20
2.2.3 Atributele HTML	20
2.3 CSS	20
2.3.1 Sintaxa	20
2.3.2 Selectorii	21
2.4 Spring Framework	21
2.5 Angular Framework	22
2.6 Node.js	23
2.6.1 Arhitectura lui Node.js	25
2.7 PostgreSQL	25
2.8 Maven	27
3 Arhitectura proiectului	29
3.1 Arhitectura generală a aplicației	29
3.2 Arhitectura aplicației Angular	29

3.3	Arhitectura aplicației Java	31
3.4	Baza de date PostgreSQL	32
4	Prezentarea aplicației	35
4.1	Introducere	35
4.2	Cerințe funcționale	35
4.3	Aplicația Angular	36
4.3.1	Descrierea aplicației	36
4.3.2	Avantajele aplicației Angular	36
4.3.3	Funcționalitățile aplicației	36
4.3.4	Meniul de navigație	38
4.3.5	Pagina de start	38
4.3.6	Pagina unde sunt afișate colecțiile	39
4.3.7	Antetul paginilor	40
4.3.8	Pagina unde sunt afișate licitațiile	41
4.3.9	Pagina unde sunt afișate informații despre proiecte în faza incipientă	41
4.3.10	Pagina unde sunt afișate colectiile care vor fi lansate .	42
4.4	Aplicatia Java	43
4.4.1	Descrierea aplicatiei	43
4.4.2	Avantajele aplicației Java	43
4.4.3	Functionalitățile aplicatiei	44
4.4.4	Directorul ”controllers”	44
4.4.5	Directorul data-transfer-object	45
4.4.6	Directorul repositories	46
4.4.7	Fisierul exceptions	47
4.4.8	Fisierul services	49
4.5	Conecțarea la PostgreSQL	50
4.5.1	Configurarea proprietatilor	50
4.5.2	Idei de dezvoltare a aplicatiei	51
5	Concluzia	52
5.0.1	Bibliografie	52

Capitolul 1

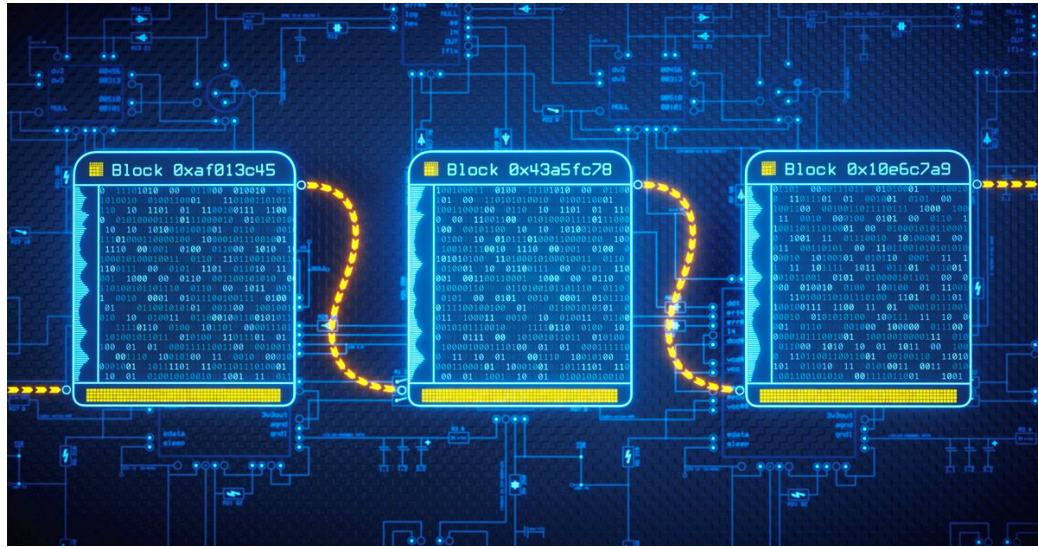
Secțiune introductivă

Tranzacționarea și colecționarea de jetoane nefungibile, abreviate NFT-uri, reprezintă cel mai recent trend din lumea cryptomonedelor. Deși acest concept există încă din 2014, ele au început să devină populare abia în anul 2020 și continuă să atragă atenția multor oameni, de la investitori la adolescenți.

Acest fenomen este în strânsă legătură cu tehnologia blockchain și cu monedele digitale, jetoanele nefungibile fiind asemănătoare cryptomonedelor, ele reprezentând active digitale și unice care nu pot fi tranzacționate pentru un alt jeton din aceeași categorie.

Ca rezultat, fiecare jeton este un produs unic, de unde și caracterul nefungibil. Transferul de jetoane de la un deținător la altul se face folosind tehnologia blockchain, ele nu având o valoare stabilită deoarece fiecare persoană este dispusă să plătească un preț diferit.

1.1 Tehnologia blockchain



Tehnologia blockchain reprezintă o bază de date distribuită ce poate fi accesată de oricine, ea nu poate fi modificată și este stocată pe calculatoarele participanților rețelei și pe multitudinea de servere din întreaga lume.

Aceasta are ca scop securizarea și înregistrarea tranzacțiilor și a datelor, fiind folosită în diverse domenii care au nevoie de un astfel de sistem.

Un blockchain conține liste de date aflate în continuă creștere, numite blocuri, care sunt legate între ele și securizate prin prisma criptografiei. Ca arhitectură, blockchain-ul reprezintă o listă simplu înlanțuită unde legăturile între elemente se fac prin intermediul hash-ului, un identificator unic.

Un bloc este alcătuit din următoare componente:

O legătură către un bloc anterior - hash-ul blocului anterior;

Un "timestamp" - data când a fost creat și adăugat un bloc în blockchain;

Date despre tranzacțiile efectuate pe blockchain;

Tranzacțiile reprezintă structuri de date care codifică transferul de valoare între participanții rețelei.

Blocurile au capacitate limitate de stocare și, atunci când își ating capacitatea maximă, sunt închise și legate de blocul umplut anterior, formând astfel un lanț de date, de aici venind și numele acestei tehnologii.

Procesul de "hashing" generează un identificator unic, un sir aleator de litere și cifre, prin combinarea valorii înregistrării anterioare cu cea a înregistrării curente într-un proces matematic, unidirecțional și ce se folosește de criptare pentru a crea această nouă valoare hash.

Hashing-ul este unidirecțional deoarece nu există un proces matematic de verificare pentru a transforma noua valoare generată înapoi la datele sale originale. În bazele de date convenționale datele pot fi alterate, pentru acestea calculându-se noi valori hash care pot fi injectate în blocurile sau înregistrările următoare pentru a se ascunde urmele.

Acest lucru nu se poate întâmplă într-o bază de date distribuită fiindcă persoana în cauză ar trebui să schimbe în mod simultan toate copiile bazei de date, ele fiind stocate pe calculatoarele participanților rețelei și existând câte o copie a acesteia pe fiecare nod de rețea.

Drept urmare sistemul blockchain este unul foarte securizat, prevenind astfel manipularea datelor. Până în prezent nici un blockchain nu a fost compromis.

Tehnologia face datele să fie private, permanente și verificabile. Crearea unui nou bloc și conectarea acestuia la lanțul existent de blocuri este un proces ireversibil, datele neputând fi manipulate, șterse sau duplicate.

O altă caracteristică importantă a acestei tehnologii o reprezintă transparență, multe blockchain-uri având registrii publici, iar istoricul tranzacțiilor execuțiate pe blockchain poate fi verificat de oricine.

Primul concept de blockchain apare prima dată în anul 2008, fiind dezvoltat de o persoană sau de un grup de persoane anonime și identificate prin pseudonimul de Satoshi Nakamoto. Odată cu apariția acestei tehnologii s-a rezolvat problema descentralizării.

Deoarece integritatea datelor nu se bazează pe o autoritate centrală, acestea neputând fi alterate vreodată, acest sistem este unul imediat folosit în domeniul medical pentru a verifica digital identitatea unui pacient, istoricul rețetelor unui pacient, date genetice sau pentru a urmări traseul comercial al medicamentelor, acesta fiind doar unul dintre multele domenii de aplicabilitate.

Avantajele unui sistem blockchain sunt următoarele:

Descentralizare:

Prin dezvoltarea conceptului de blockchain s-a rezolvat problema descentralizării, această tehnologie neavând nevoie de un grup de persoane sau instituții care să o guverneze;

Unicitate:

Activele digitale sunt distribuite și nu duplicate sau transferate, rezultând o înregistrare imuabilă a activului. Activul este descentralizat, permitând transparență publică completă și acces în timp real;

Transparentă:

Integritatea documentului este menținută printr-un jurnal transparent al modificărilor, care stimulează încrederea în active;

Securitate:

Datorită caracteristicilor de securitate încorporate și registrului public, blockchain este o tehnologie de vârf pentru practic toate industriile;

1.1.1 Arhitectura unui sistem blockchain

Principalele componente ale unui sistem blockchain sunt:

Noduri – constituie entități din cadrul rețelei ce dețin întotdeauna o copie constant actualizată a întregului blockchain;

Tranzacții – reprezintă operațiunile efectuate asupra datelor din cadrul rețelei. Acestea includ informații precum sursa, destinația și datele care sunt vehiculate între ele;

Blocuri – structuri de date care conțin detalii despre tranzacții și care sunt distribuite înlănțuit în cadrul rețelei;

Lanț – secvența de blocuri stocată pe blockchain;

Mineri – noduri specializate care utilizează puterea computațională pentru a verifica blocurile de tranzacții care urmează să fie adăugate în blockchain;

Protocol – set de reguli utilizate pentru gestionarea modului de funcționare al unei rețele;

Deoarece prin această tehnologie se poate transmite valoare oriunde în lume, într-un mod rapid și anonim, cea mai cunoscută utilizare a blockchain-ului, și poate cea mai controversată, o reprezintă cryptomonedele.

1.2 Cryptomonede



Acstea reprezintă de fapt sisteme descentralizate care se bazează pe tehnologia blockchain, proprie sau nu, fiind folosite drept mijloc de plată sau resursă de investiții.

Acstea se diferențiază de mijloacele tradiționale de plată deoarece ele funcționează pe rețele distribuite, utilizatorul putând transmite aceste monede oriunde în lume fară a fi nevoie de un intermediar.

După dezvoltarea conceptului de blockchain, în 2009 tot Satoshi Nakamoto a creat "blocul genesis" și a pornit rețeaua numită Bitcoin pe care se tranzacționează și prima cryptomonedă vreodata inventată ce are același nume.

Rețeaua s-a bucurat de un succes enorm deoarece nu a fost susținută doar de oamenii care urmăreau să facă un profit de pe urma ei, ci și de multitudinea de persoane ce credeau că aceasta nouă tehnologie este una revoluționară și ar putea ajunge un mijloc de plată folosit la nivel global.

Astfel că moneda "Bitcoin" are că scop inițial plățile efectuate online, nefiind nevoie de introducerea datelor personale sau confidențiale, de exemplu datele de pe un card. Aceasta cryptomoneda ajunge să atingă valoarea maxima de aproape 69,000 de dolari, un "Bitcoin" valorând inițial nu mai mult de un cent.

Aceasta tehnologie a deschis noi orizonturi deoarece pe succesul acestei monede a început să se contureze ideea de bani digitali, în prezent existând zeci de mii de cryptomonede, fiecare satisfăcând anumite nevoi specifice.

Acești bani digitali se împart în două categorii, cryptomonede dacă au blockchain propriu sau jetoane, în engleză "tokens", dacă nu au tehnologie

proprie și funcționează pe un blockchain existent, însă generic toate se numesc cryptomonede.

Avantajele cryptomonedelor:

Securitate:

Tehnologia blockchain face furtul mult mai greu, deoarece fiecare cryptomeda are propriul său număr de identificare unic, acesta fiind atașat proprietarilor.

Transfer rapid oriunde în lume:

Cu tehnologia blockchain, cryptomonedele pot fi transmise oricui și oriunde în lume, aproape instant și fără a fi nevoie de schimb valutar sau interferență din partea băncilor, ceea ce minimizează nevoia de bănci centrale.

Oportunitate financiară:

Oamenii pot deveni bogăți prin cryptomonede, văzute drept investiții. Unii primitori au devenit miliardari că urmare a speculațiilor care au condus la creșterea prețului cryptomonedelor, în special a monedei Bitcoin.

Valoarea unei monede se schimbă constant în funcție de anumiți factori externi, ceea ce poate reprezenta o oportunitate financiară. Această valoare poate crește datorită dezvoltării tehnologiei pe care funcționează cryptomeda, a reglementărilor favorabile, a percepției pozitive pe care o are publicul asupra ei sau a noilor soluții de afaceri care pot crește cererea monedei respective.

La fel de bine o monedă se poate devaloriza ca reacție la problemele tehnologice, manipularea pieței, atenția negativă primită din partea mass-media, eșecurile ecosistemelor pe care funcționează sau alte motive pentru care oamenii încep să își piardă încrederea în monedă și să își lichideze aceste bunuri.

Tocmai din această cauză tranzacționarea de bunuri digitale reprezintă un risc ridicat deoarece sunt mulți factori care pot influența valoarea unei cryptomonede, existând posibilitatea de a pierde banii investiți.

Un caz recent îl reprezintă cryptomeda "Luna" care de la valoarea de 86 dolari ajunge să valoreze 0,005 dolari, având o prăbușire de peste 99 la sută a valorii în doar câteva zile.

Ceea ce este interesant îl reprezintă faptul că această cryptomeda era considerată una dintre investițiile cu risc redus, atingând în trecut valoarea maximă de aproape 120 dolari, iar în prezent valorând aproape 2 dolari.

În imaginea de mai jos se regăsește graficul acestei cryptomonede care ilustrează evoluția ei de-a lungul timpului.



1.2.1 Moneda românească Elrond

Cryptomoneda denumită “Elrond” este o monedă creată de 3 români care trăiesc în Sibiu. Aceasta a luat naștere în anul 2017, fiind fondată de către Beniamin Mincu, Lucian Mincu și Lucian Todea, însă a devenit cunoscută la nivel mondial în primăvara anului 2020 când a beneficiat de o creștere de peste 5000 la sută a valorii sale de atunci.

La momentul actual, moneda virtuală este una dintre cele mai inovative de pe piață și vine cu o mulțime de avantaje comparativ cu alte cryptomonede existente.

Elrond (simbol EGLD) este prima companie românească de blockchain care a reușit să ajungă în top 50 în ceea ce privește capitalizarea de piață la nivel mondial.

Câteva avantaje ale blockchain-ului “Elrond” sunt următoarele:

Scalabilitate mare:

Sharding este o tehnică prin care datele sunt împărțite sau sparte în mai multe bucăți mai mici și care sunt distribuite în diferite zone ale rețelei, fiecare fiind formată dintr-un număr de noduri care procesează tranzacțiile, de aici și denumirea de „shard”.

EGLD combină trei tipuri de sharding:

Network Sharding

Transactions Sharding

State Sharding

într-o soluție numită Adaptive State Sharding și prin intermediul căreia se îmbunătățește exponențial comunicarea între shard-uri.

Ca atare, se constată o creștere impresionantă a performanței prin procesare paralelă, de unde rezultă creșterea cu o valoare de 1000x mai mare a numărului de tranzacții ce pot fi executate în comparative cu blockchain-urile mai vechi, cum ar fi blockchain-ul “Bitcoin”.

Eficiență și Securitate:

Elrond vine cu un protocol nou intitulat Secure Proof of Stake sau prescurtat “SPoS”, fiind un mecanism ce menține sincronizate toate computerele ce rulează pe blockchain-ul rețelei “Elrond”.

În felul acesta se asigură o latență mult mai redusă a proceselor care au loc între utilizatorii rețelei, dar și o securitate mult mai bună a sistemului.

Elrond Virtual Machine:

Acesta reprezintă un motor de inteligență artificială ce execută contractele virtuale, ajutând utilizatorii să își scrie propriile contracte inteligente în orice limbaj de programare pe care îl cunosc.

Moneda Elrond este concepută să devină un stâlp al aplicațiilor descentralizate, indiferent dacă acestea sunt legate de finanțe sau de alte nevoi ale oamenilor.

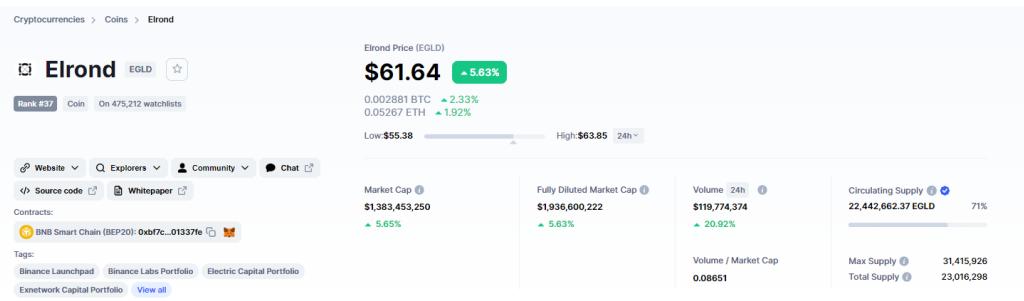
Datorită scalabilității, aplicațiile vor putea face față creșterilor semnificative de volume, iar prin combinarea tipurilor de sharding vor putea menține ritmul tranzacțiilor chiar și pe măsură ce crește volumul de cereri.

Cryptomoneda EGLD este deja folosită pe o mulțime de rețele pentru realizarea de tranzacții, drept recompense de validare sau pentru executarea de contracte inteligente.

Aprovizionarea acesteia este de 31,415,926 cryptomonede dintre care 23,016,298 sunt puse în circulație în momentul actual. Valoarea ei curentă este de 62 dolari, atingând acum câteva luni valoarea maximă de peste 520 dolari atunci când a fost lansată propria lor aplicație pe blockchain denumită “Maiar”.

Cu ajutorul ei se dorește realizarea de plăți online într-o manieră sigură datorită sistemului criptografic de care dispune, cât și securizarea rețelei și a valorii monedei prin introducerea opțiunii de “staking”, prin care investitorii își pot bloca activele digitale pentru o perioadă de timp definită în schimbul obținerii unor sume mai mari decât cele blocate.

Mai jos sunt atașate imagini care prezintă detaliile generale, cât și situația în care se află această cryptomonedă românească la momentul actual, valoarea ei raportându-se la dolarul american.



Elrond to USD Chart



1.3 Jetoane nefungibile

Jetoanele nefungibile sunt active criptografice bazate pe tehnologia blockchain, având coduri unice de identificare și metadate care le separă unele

de altele.

Ele nu pot fi tranzacționate sau schimbată pentru echivalentă, spre deosebire de cryptomonede.

Acestea sunt percepute ca fiind bunuri digitale, jetonul reprezentând de fapt accesul la acel bun digital.

Caracterul non-fungibil presupune că fiecare jeton este și are trăsături unice, fiind imposibil de replicat. Acestea sunt în număr finit, fiecare jeton face parte dintr-o anumită colecție și se generază aleator în funcție de trăsăturile și de protocolul oferit de dezvoltatorii proiectului.

Înțial ele nu au avut nici o valoare, prima colecție creată nu era privită nicidcum drept o oportunitate financiară, ea fiind distribuită gratuit printre oamenii doritori să facă parte dintr-o comunitate online pe care să o și dezvolte împreună, fiecare jeton reprezentând atât dreptul de acces în acea comunitate, cât și imaginea deținătorului în spațiul virtual.

Câțiva ani mai târziu fiecare jeton nefungibil din această colecție, denumită "Crypto Punks", ajunge să valoreze nu mai puțin de un milion de dolari pe bucătă, poate chiar și mai mult în funcție de proprietăți și trăsături deoarece ele sunt clasificate pe rarități și au șanse diferite de selecție atunci când sunt jetoanele sunt generate, drept urmare anumite NFT-uri sunt mai valoroase decât altele din aceeași colecție.

Câteva exemple de jetoane non-fungibile din colecția "Crypto Punks" sunt ilustrate în poza de mai jos, precum și unele dintre cele mai mari vânzări înregistrate:



Largest Sales

[See all top sales](#)



Acstea pot fi tranzacționate și nu au o valoare reală deoarece există doar pe blockchain, prețul fiind stabilit atât de fiecare posesor în parte, cât și de anumiți factori externi.

Acste jetoane au o altfel de valoare, valoarea lor este una perceptă și este influențată de percepția publică, mass-media sau beneficiile pe care le aduc aceste jetoane.

Ele ajung să devină cunoscute deoarece artiștii încep să își vândă creațiile, fie că este vorba de picturi, sculpturi sau muzică prin intermediul acestora. După cum vom vedea în continuare, acest fenomen are potențial mare de a revoluționa multe domenii, cât și viață noastră în mediul digital.

Fiind o tehnologie într-o continuă dezvoltare, NFT-urile încep să aibă utilitate atât în viața de zi cu zi, cât și în mediul online. Pe lângă dreptul de acces într-o comunitate sau asupra unui bun fizic, aceste jetoane încep să aibă aplicații în diverse domenii, unele devenind chiar forme de venit pasiv.

În anumite situații, NFT-urile pot fi privite drept acțiuni, dezvoltatorii proiectului răspărțind investitorii în funcție de numărul de jetoane deținute din acea colecție.

Acest lucru reprezintă un câștig pentru ambele părți deoarece o colecție devine mai valoroasă și atrage atenția investitorilor atunci când numărul de jetoane puse spre vânzare din respectiva colecție este cât mai limitat posibil, făcând-o de altfel o colecție mai rară.

Cele mai populare blockchain-uri pe care se tranzacționează jetoanele nefungibile sunt blockchain-ul Ethereum, Solana, Avalanche, Near și Elrond.

1.4 Internetul

Internetul reprezintă o rețea globală formată din calculatoare interconectate ce comunică și schimba informații între ele. Internetul este foarte des utilizat în zilele noastre în scopuri diferite, de la divertisment la informare, majoritatea oamenilor având acces la acesta.

Toate echipamentele legate la Internet, fie că este vorba de calculatoare, telefoane mobile sau alte dispozitive, comunică prin intermediul unor protocoale, ele fiind un set de reguli predefinite ce stabilesc când, cum și care este ordinea în care se execută anumite evenimente.

Fiecare dispozitiv îi este atribuită o adresă numită Protocol IP, cu ajutorul acesteia putem identifica și găsi ce dispozitiv este conectat la internet dintre miliardele de astfel de dispozitive deja conectate.

Internetul este o rețea globală de legături fizice, inclusiv fire telefonice de cupru, cabluri de televiziune și cabluri de fibră optică. Chiar și tehnologiile wireless precum Wi-Fi și 3G/4G/5G folosesc cablurile fizice pentru a se putea conecta la Internet.

Când un site web este accesat, computerul face o solicitare către un server prin intermediul cablurilor.

Site-urile web sunt păstrate pe un server ce funcționează similar cu un hard disk, stocând date despre aceste site-uri web. Serverul preia site-ul web și livrează datele înapoi către utilizator după primirea cererii.

Initial acesta era destul de limitat și principalul său scop era de a putea distribui diferite fișiere și documente, însă datorită avansului tehnologiei Internetul oferă suport pentru o grămadă de servicii, de la divertisment la cumpărături online.



1.4.1 World Wide Web

Un moment important în istoria internetului îl reprezintă anul 1991 deoarece a luat naștere tehnologia World Wide Web ce se prescurtează WWW, dezvoltarea conceptului fiind condusă de cercetătorul Tim Berners-Lee.

Acesta constituie totalitatea site-urilor, documentelor și a informațiilor de tip "hypertext" legate între ele și care pot fi accesate prin rețea de Internet.

WWW este numai unul din numeroasele servicii și aplicații informatiche disponibile pe această rețea, alte exemple de servicii fiind formulare online, coșuri de cumpărături, procesare de text, editare foto și video, conversii de fișiere, scanare de fișiere, poște electronice sau diferite posturi de radio și televiziuni.

Documentele care sunt stocate pe diverse calculatoare server pot fi regăsite prin intermediul unui identificator numit URL.

Hypertext-ul, inclusiv imagini, fișiere video și audio, este afișat cu un ajutorul unui program de navigare în web numit browser, care descarcă paginile web de pe un server și le afișează pe un terminal „client” la utilizator.

Blocurile de construcție ale Web-ului sunt pagini web formatare în limbajul HTML, conectate prin link-uri numite „hypertext” sau „hyperlink” și accesate prin HTTP. Aceste link-uri constituie conexiuni electronice care leagă informații conexe, astfel încât utilizatorii să poată accesa rapid informația dorită.

Această rețea mondială se bazează pe următoarele trei tehnologii:

HTML - limbaj de marcare hypertext;

HTTP - Protocol de transfer de hypertext;

Servere Web și browsere Web;

1.4.2 Aplicațiile Web

Odată cu dezvoltarea conceptului de World Wide Web apar și aplicațiile web, ele fiind în strânsă legătură cu acest serviciu și se folosesc de browser-ele web pentru a efectua diferite sarcini pe Internet.

Aplicațiile Web permit utilizatorilor să interacționeze cu o companie sau cu persoanele în cauză, indiferent de distanță sau de dispozitivele folosite.

Aplicațiile web necesită un server web pentru a gestiona cererile clientului, un alt server de aplicații pentru a efectua sarcinile solicitate și o bază de date pentru a putea stoca anumite informații.

Interacțiunea generică dintre un client și o aplicație web arată în felul următor:

1. Utilizatorul declanșează o cerere către serverul web prin Internet, fie printr-un browser sau prin interfață de utilizator pe care o oferă aplicația.

2. Serverul web transmite această cerere serverului de aplicații web corespunzător.

3. Serverul de aplicații web execută sarcina cerută, de exemplu prelucrarea datelor sau interogarea bazei de date, generând un rezultat pe baza datelor solicitate.

4. Serverul de aplicații web trimit rezultatele obținute către serverul web.

5. Serverul web răspunde înapoi la client cu datele prelucrate.

1.5 Scopul proiectului

Proiectul a pornit de la dorința de a simula o platformă online unde oamenii pot tranzacționa sau a pune la licitație jetoane nefungibile existente pe blockchain-ul "Solana", acest blockchain fiind un sistem unde valoarea este transmisă oriunde în lume și într-un mod rapid, costurile tranzacției fiind aproape inexistente.

De asemenea, aceasta platformă reprezintă și un mijloc prin care oamenii își pot promova propriile proiecte sau creații artistice prin tranzacționarea jetoanelor din acele colecții sau prin licitarea lor.

Un factor important pentru a garanta succesul unui proiect îl reprezintă promovarea într-un mediu adecvat și format din oameni puși în temă cu acest domeniu.

Indiferent de funcționalitățile noi aduse, lipsa unei promovări corespunzătoare și a unui public specific scad șansele ca un proiect să devina unul de succes.

1.5.1 Obiectivele proiectului

Obiectivele generale ale acestei platforme de tranzacționare pot fi formulate astfel:

Tranzacționarea de jetoane din colecții deja existente;

Posibilitatea de a pune un jeton la licitație, cât și de a participa la alte licitații;

Promovarea și explorarea colecțiilor existente, cât și a celor în faza incipientă și care urmează să fie lansate;

Promovarea acestui fenomen deoarece tehnologiile apărute ca rezultat al acestuia au revoluționat deja diverse domenii, lumea fiind într-o continuă digitalizare;

Posibilitatea de a urmări situația actuală a pieței printr-un tabel de statistici;

Posibilitatea creatorilor de a-și lansa proiectele prin intermediul platformei;

Capitolul 2

Tehnologiile folosite

2.1 Java

Java este un limbaj de programare cu scop general și care a fost proiectat să funcționeze pe o varietate largă de dispozitive, de la calculatoare și telefoane de tip smartphone până la televizoare smart.

Acest limbaj este unul puternic tipizat și capabil de programare orientată pe obiecte, ceea ce înseamnă că permite organizarea și gruparea atributelor sub formă de obiecte care aparțin unor clase.

Odată ce un obiect este definit, acesta poate fi referențiat mai târziu în cod fară a fi nevoie de a îl redefini.

Java dispune de o interfață de programare a aplicațiilor foarte bogată, precum și de un ecosistem cu sursă deschisă, acest limbaj având o multime de unelte integrate care ajută programatorul în dezvoltarea aplicatiei sale.

Acest limbaj este unul extrem de versatil și poate fi găsit aproape în orice industrie, fiind folosit pentru programarea de aplicații mobile, web, desktop sau de tip enterprise.

Java este mai mult decât un limbaj de programare, el reprezentă de fapt o platformă și fiecare versiune a să este proiectată conform unor cerințe specifice.

De exemplu Java Enterprise este o versiune specială a acestei platforme concepută pentru aplicațiile de tip business, fiind folosită pentru rețele cu scara largă și venind cu un nivel de securitate mai înalt.

Cu acest limbaj de programare se pot realiza aproape orice fel de aplicații, excepție făcând cele „low level” cum ar fi componente ale sistemului de operare, drivere și programele unde este nevoie de procesare de performanță în timp real.

Java a fost proiectat ținând cont de următoarele principii:

Ușurință de utilizare:

Fundamentele Java provin dintr-un limbaj de programare numit C++, deși acesta este un limbaj puternic și complex în sintaxă, este inadecvat pentru anumite cerințe. Java a preluat și îmbunătățit ideile C++ pentru a oferi un limbaj de programare puternic, simplu de utilizat și care să fie adecvat pentru noile cerințe de pe piață;

Fiabilitate:

Java este un instrument bun pentru a reduce probabilitatea de erori fatale generate de greșelile programatorului. În acest sens, a fost introdusă notiunea de programare orientată pe obiecte, datele fiind împachetate și manipulate împreună într-un singur loc, în obiecte ale claselor;

Securitate:

Deoarece Java vizează programarea pe dispozitive mobile care fac schimb de date prin intermediul rețelelor, acesta a fost construit pentru a include un nivel ridicat de securitate, fiind de altfel cel mai securizat limbaj de programare;

Independență platformei:

Programele trebuie să funcționeze indiferent de mașinile pe care sunt executate. Java a fost scris pentru a fi un limbaj portabil și multi-platformă, necontand sistemul de operare, componente hardware sau dispozitivele pe care rulează;

2.2 HTML

HyperText Markup Language sau "HTML" constituie un limbaj de marcă folosit în construcția paginilor web afișate prin intermediul browserului.

Folosind elemente, etichete și atribute, acesta permite utilizatorilor să proiecteze și să organizeze secțiuni, paragrafe și conexiuni.

Printre cazurile de utilizare se numără următoarele:

Dezvoltare web:

Dezvoltatorii folosesc codul HTML pentru a proiecta modul în care un browser afișează elementele paginii web, cum ar fi text, hyperlinkuri și fișiere media.

Navigare pe internet:

Utilizatorii pot naviga cu ușurință și inseră legături între paginile consecutive și site-urile web, deoarece HTML este foarte folosit pentru a încorpora hyperlinkuri.

Documentație web:

HTML face posibilă organizarea și formatarea documentelor.

2.2.1 Etichetele HTML

Un tag reprezintă o secvență de cod cuprinsă între simbolurile ”`<`” și ”`>`” și este folosit pentru a specifica regiuni ale documentului HTML, acestea fiind interpretate ulterior de browser.

2.2.2 Elementele HTML

Elementele sunt formate din cuvinte pereche ce marchează deschiderea și închiderea acestuia împreună cu informațiile din interiorul elementului.

2.2.3 Atributele HTML

Atributele sunt folosite pentru a modifica și schimba aspectul elementelor HTML ce urmează să fie interpretate de browser, acestea putând avea un număr nelimitat de atribute.

2.3 CSS

Limbajul de programare CSS este una dintre tehnologiile de bază pentru a construi pagini Web și este folosit pentru a descrie și contura un document scris în alte limbaje de programare, de exemplu HTML.

Acest limbaj permite individualizarea și separarea codului, cu scopul de a modifica aspectul elementelor din paginile Web, precum spațierea, așezarea în pagină și a modului cum sunt colorate sau înfățișate.

Individualizarea sporește accesul asupra elementelor și oferă o flexibilitate sporită asupra specificațiilor și cerințelor de înfățișare ale aplicației.

2.3.1 Sintaxa

Limbajul CSS se bazează pe o sintaxă simplă și utilizează o serie de cuvinte cheie în engleză pentru a stabili caracteristicile de modificare și alterare al conținutului.

O pagină de stil conține o serie de reguli formate din mai mulți selectori și blocuri declarative.

2.3.2 Selectori

În acest limbaj de programare selectorii sunt folosiți pentru a declara trăsătura sau trăsăturile de stil care vor fi aplicate asupra unui element prin intermediul atributelor definite.

Aceștia pot fi aplicati pe toate elementele de tip specific prin intermediul atributelor.

Un exemplu de atribut îl reprezintă cuvântul "id" care este recunoscut prin prefixul "hash" și are rol de identificator unic.

2.4 Spring Framework

Spring Framework reprezintă un cadru de aplicații open source care oferă un suport vast pentru crearea unei infrastructuri de dezvoltare a aplicațiilor Java, astfel programatorul rămânând concentrat asupra funcționalității.

La nivel de funcționalitate, acest framework reprezinta un container de injectie de dependințe

Spring framework este împărțit în peste 20 de module, fiind divizate în grupuri în funcție de scopul acestora.

Spring Boot constituie o extensie a acestei platforme care permite eliminarea configurațiilor de nivel inferior, acestea fiind făcute automat prin alegerea unor opțiuni de către programator în momentul inițierii aplicației.

Java Spring Boot este folosit la crearea de micro-servicii, arhitectură ce permite dezvoltatorilor să creeze servicii independent, fiecare serviciu având procesul lui propriu. Spring Boot permite accelerarea dezvoltării unei aplicații web.

Această extensie analizează configurațiile setate de către programator și injectează toate dependențele necesare pentru satisfacerea cerințelor.

Câteva dintre avantajele oferite de această tehnologie sunt autoconfigurarea, independența și autonomia acestuia de a lua decizii astfel încât să optimizeze cât mai bine aplicația. Spring Boot este un modul independent și oferă posibilitatea de a integra toate serviciile automat.

De exemplu Spring Boot nu are nevoie de un server container în care să fie plasată aplicația deoarece acesta oferă unul implicit pe care îl rulează la startul aplicației.

Tehnologia este amplasată în vârful ierarhiei Spring Framework, ea fiind capabilă să furnizeze toate funcționalitățile acestui cadru de lucru pe care le și integrează în proiect, scutind programatorul de configurarea infrastructurii și oferind mai mult timp pentru dezvoltarea funcționalului de baza.

2.5 Angular Framework

Angular este o platformă dezvoltată de Google și care este folosită pentru construirea de Single Page Applications în HTML, CSS și TypeScript. TypeScript reprezintă un limbaj modern, curat și care vine drept o îmbunătățire a limbajului JavaScript prin implementarea de noi funcționalități, orice cod TypeScript fiind compilat în cod JavaScript pentru a putea fi înțeles de browser.

Această platformă de dezvoltare include următoarele:

Un cadru bazat pe componente pentru construirea de aplicații web scalabile;

O colecție numeroasă de biblioteci bine întregate ce constituie soluții pentru anumite sarcini specific, de exemplu gestionarea formularelor, rutarea sau comunicarea dintre client și server;

O suită de instrumente menite să ajute dezvoltatorii pentru a construi, testa și actualiza codul;

Framework-urile îmbunătățesc eficiența și performanța dezvoltării web, oferind un cadru consistent care elimină nevoia dezvoltatorilor de a rescrie cod de la zero.

Acestea economisesc timp și oferă dezvoltatorilor o mulțime de funcționalități suplimentare care pot fi adăugate la software cu un efort minim.

Unul dintre avantajele folosirii framework-ului Angular îl reprezintă arhitectura să bazata pe componente.

Componentele unghiulare sunt elemente mici ale interfeței cu utilizatorul, similară cu secțiunile unei aplicații.

În Angular, există o ierarhie strânsă a componentelor, chiar dacă fiecare componentă este conținută cu propria să funcționalitate. Fiecare element UI de pe ecran poate fi o componentă în Angular.

Avantajele folosirii acestui cadru de lucru sunt:

Siguranță:

Puteți considera Angular că fiind un framework de încredere deoarece este dezvoltat de Google;

Timp de dezvoltare redus:

Angular se bazează pe mașina virtuală JavaScript actuală transformând şabloanele în cod. Timpul de încărcare al lui Angular este, de asemenea, rapid;

Facilitează testele unitare:

Legarea de date bidirecțională a modulelor și componentelor constituie o caracteristică a acestui cadru de lucru ce face codul să fie consistent și ușor de înțeles pentru testarea unitară;

Pe parcursul procesului de dezvoltare al aplicației, fiecare unitate de cod este verificată independent, asigurând astfel un control riguros al calității.

Multi-platformă:

PWA-urile construite cu Angular pot rula pe o gamă largă de dispozitive, iar cadrul este utilizat pe scară largă în aplicațiile mobile native. Anterior, dezvoltatorii front-end foloseau o combinație multiplatformă între Ionic și Angular. Cea mai populară asociere a sa în zilele noastre este cu NativeScript;

Comunitate:

Angular oferă o comunitate și un ecosistem minunat care este bine susținut. Există o mulțime de conținut pe cadru, inclusiv sub formă de instrucțiuni și videoclipuri, precum și o serie de instrumente terțe;

Câteva exemple de companii de ultimă oră ce se folosesc de acest cadru de lucru în conducta lor modernă de dezvoltare a aplicațiilor web sunt: Google, Gmail, Paypal, Samsung sau Weather.com.

2.6 Node.js

Node.js este un cadru JavaScript care poate rula pe o varietate de sisteme, inclusiv Linux, Mac OS, Unix și Windows.

Permite dezvoltatorilor să utilizeze JavaScript pentru a rula scripturi pe server, cu scopul de a produce conținut dinamic al paginii înainte că această să fie trimisă la browser.

Are un design bazat pe evenimente, cu capacitați de intrare și ieșire asincrone (I/O). Această opțiune de proiectare își propune să îmbunătățească producția și scalabilitatea în aplicațiile care necesită operațiuni I/O multiple sau aplicații în timp real.

Node.js poate gestiona mii de conexiuni simultane cu un singur server fără a adăuga supraîncărcarea de gestionare a concurenței firelor, reprezentând o sursă majoră de erori.

Astfel că acest cadru de dezvoltare este ideal pentru crearea de aplicații cu consum mare de date, scalabile și care sunt orientate pe partea de server în general.

Câteva dintre avantajele utilizării lui Node.js sunt:

Dezvoltare rapidă:

Aplicațiile sunt construite pe motorul JavaScript Google V8 care include biblioteca Node.js, fiind este extrem de rapid la rularea codului;

Fără stocare în tampon de date:

Datele nu sunt stocate în tampon în aplicațiile Node.js;

Bazat pe evenimente și asincron:

Fiecare interfață de programare a aplicațiilor (API) din biblioteca Node.js este asincronă și se bazează pe evenimente. Se precizează în mod explicit că este un cadru neblocant.

Un server Node.js nu regrezează datele în timp ce așteaptă că un API să facă acest lucru. Astfel de modificări ale serverului la următoarele API-uri și sisteme de notificare a evenimentelor ajută serverul să primească un răspuns de la apelul API anterior.

Licență:

Node.js este publicat sub licență MIT și este accesibil tuturor utilizatorilor.

Un singur fir de execuție și scalabilitate:

Are un design cu un singur fir de execuție, precum și o buclă de evenimente, ceea ce îl face scalabil. Sistemul de evenimente permite serverului să răspundă fără a se bloca.

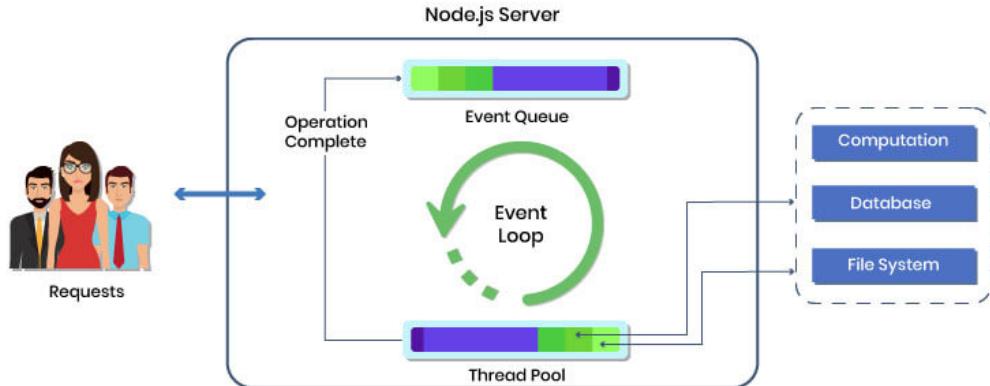
Spre deosebire de serverele tradiționale care generează fire restricționate pentru a gestiona cererile, acest lucru face serverul incredibil de scalabil.

În comparație cu serverele obișnuite, utilizează un singur model “threaded” și poate suporta un număr mai mare de solicitări cu același model.

Pentru a gestiona mai mulți consumatori concurenți, Node.js folosește o paradigmă „Single Threaded Event Loop”. Aceasta utilizează un formular JavaScript bazat pe evenimente cu o funcție JavaScript de apel invers ca paradigmă de procesare.

Mai jos este prezentata diagrama arhitecturii Node.js care indică funcția unui server construit utilizând acest cadru.

2.6.1 Arhitectura lui Node.js



Elementele unei arhitecturi de tip Node.js sunt:

Solicitări: În funcție de munca pe care utilizatorul trebuie să o completeze într-o aplicație, cererea primită poate fi clasificată ca neblocanta(simplă) sau blocanta(compliicată);

Server: Serverul este un cadru care preia cererile utilizatorilor, le procesează și apoi trimită înapoi un răspuns cu datele procesate;

Coadă de evenimente: Coada de evenimente din cadrul Node.js colectează cererile primite de la client și le trimită individual la bucla de evenimente;

Thread Pool: Aceasta conține toate fișele de execuție care pot fi utilizate pentru a îndeplini sarcinile necesare în funcție de cererile clientilor;

Buclă de evenimente: Aceasta întreține procesele și solicitările înainte de a returna răspunsurile clientilor succesorilor;

2.7 PostgreSQL

PostgreSQL este un sistem de baze de date robust, open source care utilizează și extinde limbajul SQL, având o serie de capabilități ce îi permit să stocheze și să crească în mod fiabil chiar și cele mai complexe sarcini de lucru cu date.

PostgreSQL a fost creat în 1986 ca parte a proiectului intitulat “POSTGRES” de la Universitatea din Berkeley, California, platforma de bază fiind dezvoltată activ de mai bine de 30 de ani.

PostgreSQL are o reputație solidă pentru design-ul său de încredere, integritatea datelor, setul larg de caracteristici, extensibilitatea și angajamentul comunității open source de a furniza în mod continuu soluții performante și inovatoare.

Încă din 2001, PostgreSQL este compatibil cu ACID și rulează pe toate sistemele de operare majore.

ACID (atomicitate, consecvență, izolare și durabilitate) este un set de calități ale tranzacțiilor bazei de date concepute pentru a asigura validitatea datelor în fața erorilor, întreruperilor de curent și a altor calamități.

O tranzacție este un set de activități ale bazei de date care satisfac calitățile ACID și poate fi văzută că o singură operațiune logică asupra datelor. Avantajele folosirii acestui sistem de baze de date sunt:

Tipurile de date:

Primitive: întreg, numeric, sir de caractere, boolean;

Structurate: Data/ora, Matrice, Interval;

Documente: JSON/JSONB, XML, cheie-valoare (Hstore);

Geometrice: punct, linie, cerc;

Personalizate: tipuri personalizate de date;

Integritatea datelor:

UNIQUE – unic;

NOT NULL – diferit de NULL;

Chei primare;

Chei străine;

Constrângeri de excludere;

Blocări explicite, blocări consultative;

Concurență și performanță:

Indexare: arbori B, multicolanoa, parțială;

Indexare avansată: GiST, SP-Gist, KNN Gist, GIN, BRIN, indici de acoperire sau filtre Bloom;

Planificator/optimizator de interogări sofisticat, scanări numai indexate, statistici pe mai multe coloane;

Paralelizarea interogărilor de citire și construirea de indici cu ajutorul arborilor B; Partitionarea tabelelor;

Compilare “just-in-time” (JIT) de expresii;

Fiabilitate și recuperare în caz de dezastru:

Replicare: asincronă, sincronă, logică;

Recuperare punct în timp (Point In Time Recovery);

Securitate și autentificare:

Autentificare: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, certificate; Sistem robust de control al accesului;

Securitate la nivel de coloană și rând;

Autentificare multi-factor cu certificate;

Extensibilitate:

Funcții și proceduri stocate;

Limbaje procedurale: PL/PGSQL, Perl, Python;

Expresii de cale în SQL/JSON;

Wrapper-uri pentru date străine: printr-o interfață SQL convențională se poate realiza conexiunea la alte baze de date sau fluxuri;

Internationalizare și căutare text

Suport pentru seturi internaționale de caractere;

Colațiuni care nu țin cont de majuscule și minuscule;

Căutare full-text;

2.8 Maven

Maven este un instrument de compilare open-source proeminent, creat de Apache Group ce permite construirea, publicarea și implementarea mai multor proiecte, oferind și un management mai ușor al acestora.

Dezvoltatorii pot folosi instrumentul pentru a crea și documenta un cadru de ciclu de viață.

Acest instrument de compilare Java poate fi folosit pentru a crea proiecte în C Sharp, Scala, Ruby, precum și alte limbi de programare.

Acest instrument, care se bazează pe Modelul obiect al proiectului (POM), a făcut mai ușor pentru dezvoltatori să revizuiască versiuni, să producă rapoarte și să testeze setările automate.

Dependințe de proiect, construcția de profiluri, plugin-uri sau alte obiective ce pot fi rulate sunt doar câteva exemple de configurații care pot fi descrise în POM.

Maven vine cu o serie de avantaje care explică de ce este un instrument atât de popular.

Următoarele sunt câteva dintre cele mai notabile caracteristici ale lui Maven: O bază de date masivă ce este în continuă expansiune, cu biblioteci create de utilizatori;

Abilitatea de a configura rapid proiecte, respectând cele mai bune practice;

Gestionarea dependențelor cu actualizare automată;

Se asigură că toate proiectele au același aspect;

Este ușor de extins și puteți dezvolta plug-in-uri în limbaje de scripting sau Java;

Modelul obiect al proiectului (POM), fiind un fișier XML care conține toți parametrii de proiect și de configurare, îl face pe Maven un instrument atât de valoros.

POM-ul poate fi găsit în directorul rădăcină al aplicației și conține descrierea proiectului, informații de versiune și informații de gestionare a configurației.

Capitolul 3

Arhitectura proiectului

3.1 Arhitectura generală a aplicației

Aplicația este împărțită în două sisteme care comunică între ele constant, o aplicație dezvoltată în cadrul de lucru Angular și una scrisă în limbajul de programare Java.

Aplicația dezvoltată în framework-ul Angular are ca scop interacțiunea cu clientul și constituie o interfață grafică unde datele sunt transmise în timp real.

Aplicația Java este folosită pentru a interacționa cu serverul, fiind responsabilă pentru a prelua informațiile transmise de utilizator în vederea integrării bazei de date.

Serverul trimite un răspuns înapoi cu datele procesate către interfață grafică.

3.2 Arhitectura aplicației Angular

Aplicația Angular este structurată sub 3 fișiere:

Fișierul “Components”;

Fișierul “Models”;

Fișierul “Services”;

```
✓ licenta
  > dist
  > node_modules
  ✓ src
    ✓ app
      > Components
      > Models
      > Services
      TS app-routing.module.ts M
      # app.component.css
      ◁ app.component.html M
      TS app.component.spec.ts
      TS app.component.ts M
      TS app.module.ts M
```

Directorul “Components” conține toate componentele folosite în realizarea interfeței grafice și a anumitor secțiuni din aplicație.

Directorul “Models” conține clasele necesare pentru a reprezenta în interfață jetoanele nefungibile, colecțiile, licitațiile și viitoarele colecții care sunt în faza

de dezvoltare.

Directorul “Services” conține doar servicii și este responsabil pentru popularea interfeței cu date specifice fiecărei secțiuni și interacțiunea cu baza de date.

3.3 Arhitectura aplicației Java

Aplicatia Java este structurată dupa funcționalitățile claselor sub forma a 6 fișiere:

Fisierul “controllers”:

Acest fisier gestionează clasele de controlori responsabile cu apelurile API REST primite și cu returnarea datelor către client.

Fisierul “data-transfer-object”:

Acest fisier conține clase de obiecte care vizează interacțiunea cu interfața grafică de utilizator.

Fisierul “exceptions”:

Acest fisier conține clase ce implementează excepțiile care pot apărea atunci când baza de date este integrată sau când apelurile de tip HTTP sunt testate.

Fisierul “models”:

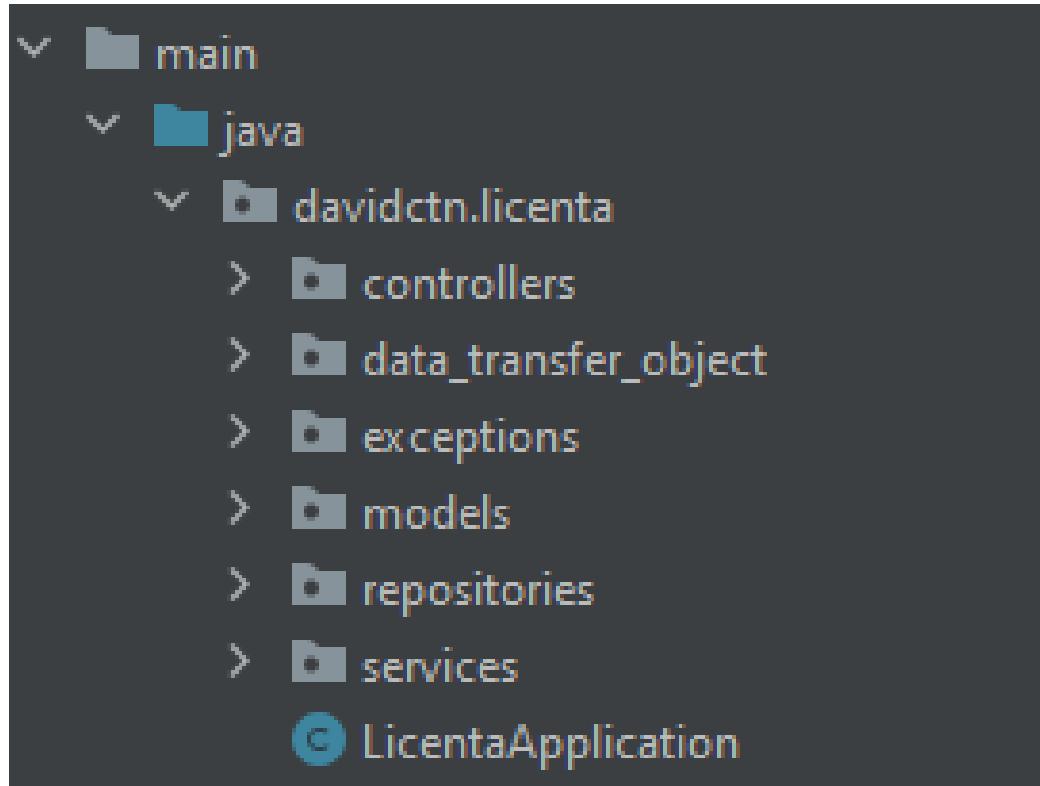
Acest fisier conține clase de obiecte care vizează interacțiunea cu baza de date și care pot fi stocate în acest mediu.

Fisierul “repositories”:

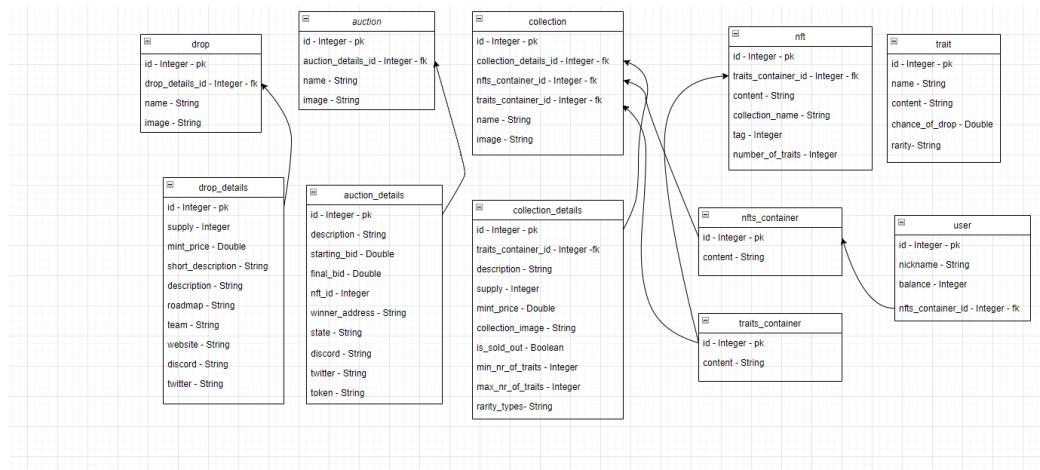
Directorul “repositories” este alcătuit din interfețe ce extind librăria JPA-Repository ce implementează metode și alte tipuri de date pentru a permite persistența în aplicație.

Fisierul “services”:

Acest fisier conține clase ce implementează metodele mapate în directorul “controllers” și care asigură persistența în aplicație.



3.4 Baza de date PostgreSQL



Normalizarea este o tehnică ce se folosește de o serie de teste, numite forme normale, pentru a putea grupa eficient atributele și, în cele din urmă,

pentru a stabili setul adecvat de relații ce susțin anumite cerințe specifice.

Tehnica de normalizare este construită în jurul conceptului de formă normală ce definește un sistem de constrângerile. Dacă o relație îndeplinește constrângerile unei anumite forme normale, se spune că relația se află în acea formă normală.

A treia formă normală a unei baze de date relaționale constituie o tehnică de proiectare a acesteia, unde toate atributele sau câmpurile lor depind direct de cheia primară, fiind eliminate posibilele ambiguități sau folosire infiecientă a memoriei.

În această bază de date tabelele modeleză obiecte precum colecții, licitații, drop-uri, utilizatori, jetoane non-fungibile, trăsături ale acestora, precum și ale containerelor specifice obiectelor menționate anterior, ele stocând informații referitoare la acel obiect.

Tabelul pentru colecții reprezinta proiectele ale căror jetoane nefungibile au fost generate și se tranzacționează pe blockchain, cele afișate în interfață reprezentând cele puse spre vânzare de deținători.

Tabelul pentru drop-uri constituie colectiile care sunt inca in faza incipientă, de dezvoltare neavand generate jetoanele non-fungibile.

Când se lansează o colecție, perioada de generare a jetoanelor este una destul de scurta, în general între 24 și 48 de ore.

Dacă o colecție are o aprovisionare de 7000 de jetoane, însă în această perioadă de timp doar 3000 de oameni aleg să investească în proiect prin achiziționarea unui jeton, după terminarea acesteia, aprovisionarea va fi redusă la 3000 deoarece doar atâția investitori au participat la lansarea publică.

Acest lucru poate avea un impact pozitiv asupra proiectului deoarece o colecție ce are o aprovisionare mai limitată, de exemplu între 500 și 1500 de bucăți, poate ajunge mai rapid o colecție percepută fiind o colecție mai rară decât altele deoarece numărul de jetoane nefungibile care poate fi pus spre vânzare este de la început mai limitat.

Procesul de generare al unui jeton, în timpul lansării publice, se numește "mint" și în general costa între 0.5-3 Solana.

Există proiectele al căror preț de lansare este mai mic 0.5 Solana, putând fi chiar și 0.005 Solana sau gratuită, fiind percepută doar taxa pe tranzacție.

Însă există și proiecte al căror preț de lansare depășește 20-30 Solana, ele reprezentând de fapt chei de acces către anumite tehnologii sau chiar acea tehnologie propriu zisă, de exemplu noduri de rețea specializate să execute anumite sarcini specifice sau instrumente avansate de analiză pentru piața cryptomonedelor.

În fiecare tabel există un câmp denumit "id" ce reprezintă cheia primară ale acestor tabele, fiind una de tip întreg și care se auto-incrementează cu o unitate atunci când un nou obiect este stocat.

Tabelele de detalii reprezintă containere pentru informații legate de anumite obiecte, precum colecții, licitații și drop-uri.

Cheile primare ale acestora reprezentate de câmpurile denumite “id” constituie chei străine în tabelele ale căror informații le prezintă.

Tabelele denumite ”container” au ca scop stocarea identificatorilor ai jetoanelor nefungibile dintr-o colecție existentă sub forma de sir de caractere, separate prin caracterul „,” și a tuturor trăsăturilor posibile dintr-o colecție pentru a se putea genera aceste jetoane.

Aceste tabele conțin chei primare reprezentate de câmpurile denumite “id” ce constituie, de asemenea, chei străine în tabelul reprezentat colecțiile de jetoane.

Capitolul 4

Prezentarea aplicației

4.1 Introducere

Acest proiect a fost inspirat din dorința de a simula o platformă unde oamenii pot tranzacționa jetoane nefungibile, pot participa la licitații, pot explora colecțiile existente pe blockchain sau proiectele în faza incipientă și pot urmări situația pietiei de jetoane. Această aplicație este formată din 3 parti:

Aplicația Angular;
Aplicația Java;
Baza de date PostgreSQL;

4.2 Cerințe funcționale

Cerințele funcționale ale aplicației pot fi formulate în felul urmator:

Afișarea jetoanelor non-fungibile puse spre vânzare în interfața utilizator;

Posibilitatea explorării unei colecții pentru a vedea ce jetoane sunt puse spre vânzare, precum și pentru a afla mai multe informații despre aceasta;

Tranzacționarea jetoanelor nefungibile;

Posibilitatea participării la licitații;

Explorarea viitoarelor colecții încă în faza de dezvoltare;

4.3 Aplicația Angular

4.3.1 Descrierea aplicației

4.3.2 Avantajele aplicației Angular

Eficiență:

Rapiditatea cu care funcționează sistemul;

Compatibilitate:

Posibilitatea de a utiliza aplicația pe un număr foarte vast de dispozitive, de la telefoane mobile la calculatoare;

Validarea datelor:

Funcționarea aplicației astfel încât în urma interacțiunii cu clientul să nu se ajungă la un comportament incorrect al aplicației;

Această aplicație Angular reprezintă o interfață prin care utilizatorul poate să vizualizeze și să exploreze colecțiile de jetoane existente în baza de date, în timp real.

Scopul aplicației este acela că utilizatorul să poată interacționa cu jetoane nefungibile într-un mod facil și printr-o interfață user-friendly. Prin acest sistem, utilizatorul poate să interacționeze cu colecțiile existente prin tranzacționarea acestor jetoane.

4.3.3 Funcționalitățile aplicației

Aplicația este proiectată în cu scopul de a oferi posibilitatea utilizatorilor să vizualizeze în timp real datele transmise în interfață.

Astfel că un utilizator poate explora colecțiile în funcție de categoriile din care fac parte, poate vinde sau cumpăra jetoane din aceste colecții, poate participa la licitații sau chiar să își pună la licitație jetoanele sale nefungibile.

De asemenea, interfața are o secțiune pentru proiectele care sunt în faza de dezvoltare, unde utilizatorii se pot informa despre ce promit să aducă proiectele.

Aplicația Angular este structurată sub 3 fișiere:

Fisierul “Components”;

Fisierul “Models”;

Fisierul “Services”;

```
✓ licenta
  > dist
  > node_modules
  ✓ src
    ✓ app
      > Components
      > Models
      > Services
      TS app-routing.module.ts M
      # app.component.css
      ◁ app.component.html M
      TS app.component.spec.ts
      TS app.component.ts M
      TS app.module.ts M
```

Directorul “Components” conține toate componentele folosite în realizarea interfeței grafice, cât și diferite secțiuni din aplicație.

Directorul “Models” conține clasele necesare pentru a reprezenta în interfață jetoanele, colecțiile, licitațiile și viitoarele colecții care sunt în faza de dezvol-

tare.

Directorul “Services” conține doar servicii și este responsabil pentru popularea interfeței și interacțiunea cu baza de date.

4.3.4 Meniul de navigație

Navigarea în aplicație se face folosind un meniu care este prezent în fiecare pagină a aplicației, el având o poziție fixă în partea stangă a ecranului.

Butonul “Home” al acestui meniu ne conduce către pagina principală a aplicației, unde putem explora colecțiile de jetoane nefungibile în funcție de mai multe criterii, de exemplu dacă acestea sunt populare, nou apărute pe piață sau se bazează pe o realitate virtuală, respectiv un joc video.

Butonul “Launchpad” ne trimit către secțiunea cu același nume unde proiectele intra în faza de lansare publică, aici generându-se jetoanele nefungibile ale colecției. Butonul “Auctions” ne trimit către secțiunea de licitații unde putem vedea ce licitații au loc în prezent, precum și informații despre licitațiile anterioare sau viitoare.

Butonul “Drop Calendar” ne trimit către secțiunea unde sunt afisate în ordine calendaristică viitoarele colecții ce urmează să fie lansate, tot aici găsind și informații despre aceste proiecte.

Butonul “Collections Stats” ne conduce către secțiunea de statistici unde utilizatorul poate urmări situația actuală a proiectelor în vederea formării unei imagini de ansamblu asupra pieței.

4.3.5 Pagina de start

Pagina de start a aplicației reprezintă pagina “Home” din meniul de navigare al aplicației. Aici se pot urmări colecțiile de jetoane non-fungibile în funcție de categoriile din care fac parte și pe ce tehnologii funcționează.

Colecțiile sunt împărțite pe următoarele criterii:

Colecții populare;

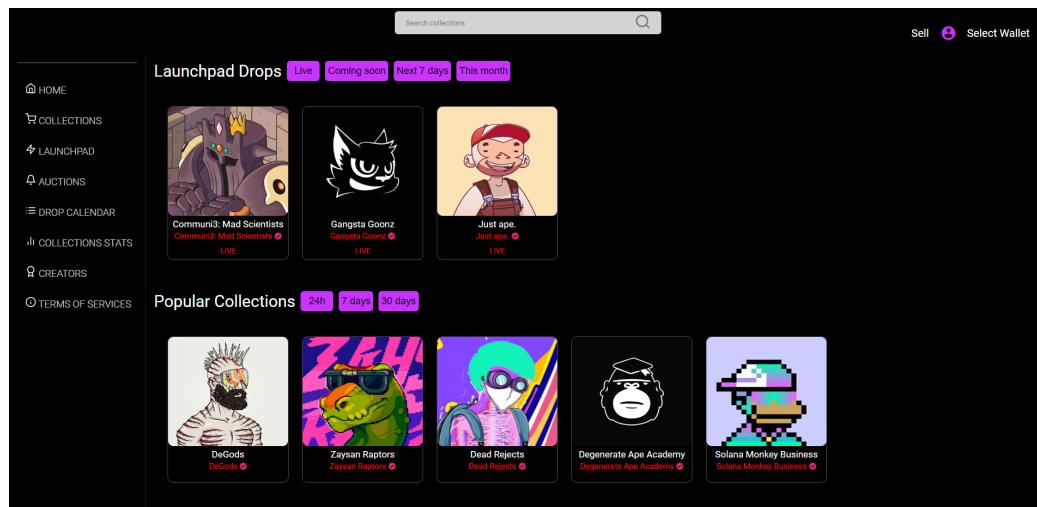
Colecții noi apărute;

Colecții ce vizează realitatea virtuală sau jocuri video, jetoanele fiind dreptul de acces către aceste realități, respectiv jocuri video;

Colecții de tip organizație autonomă descentralizată(DAO) unde direcțiile spre care se îndreaptă proiectele sunt decise de către comunitate, fiecare jeton nefungibil reprezentând un drept de vot, votul majoritar câștigând.

Tot pe pagina principală se pot urmări atât licitațiile de jetoane care se întâmplă în prezent sau care vor urma, fiind prezentate informații despre jetonul care este scos la licitație, de exemplu adresa jetonului, din ce colecție face parte sau care este valoarea curentă la care este licitat.

Pagina principală este împărțită pe mai multe secțiuni și fiecare corespunde unor tipuri de colecții sau licitații. La apăsarea acestora, în funcție de secțiune, utilizatorul este redirectionat către colecția sau licitația respectivă.

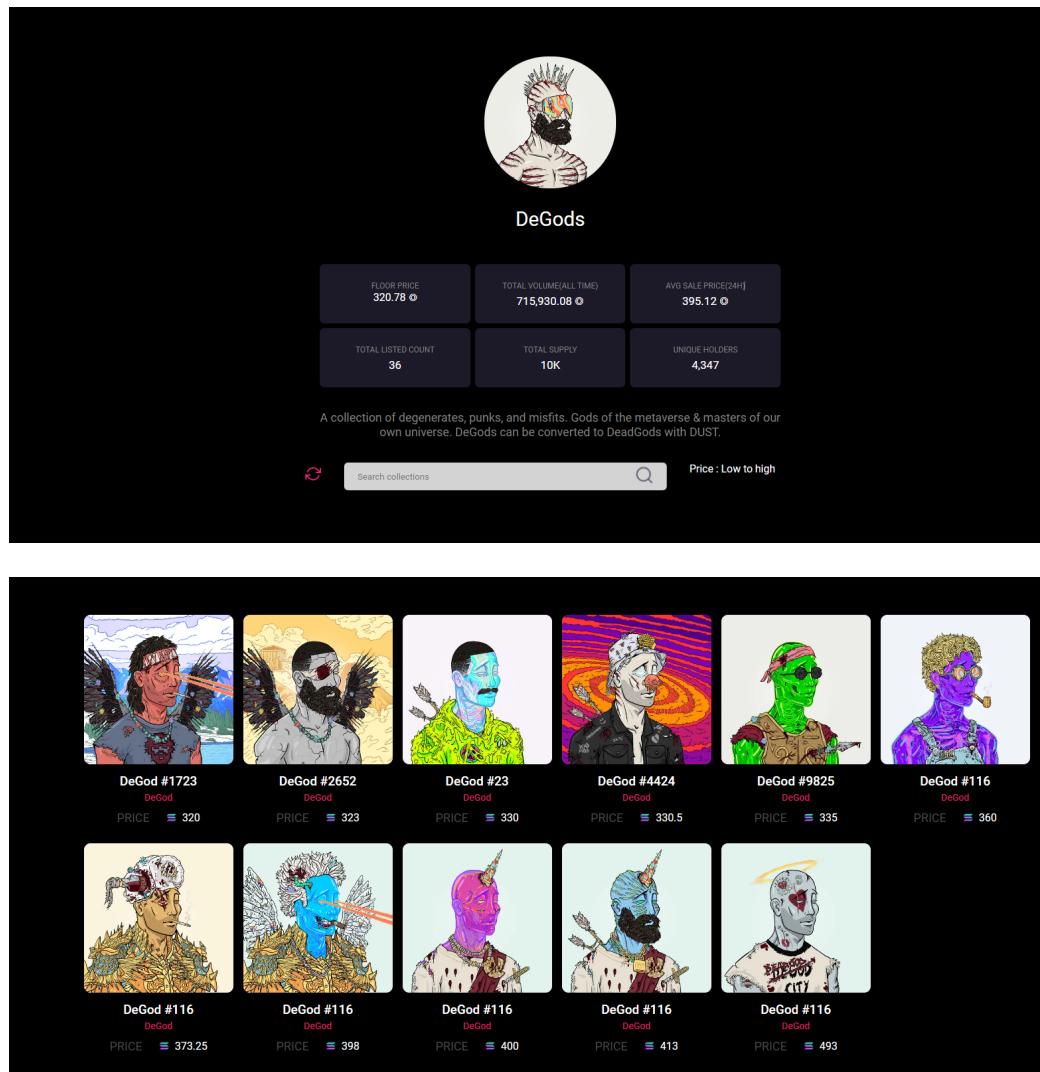


4.3.6 Pagina unde sunt afișate colecțiile

Dacă utilizatorul alege o colecție deja existentă, acesta este redirectionat către o pagină unde poate vedea jetoanele nefungibile puse spre vânzare din acea colecție, cât și câteva informații referitoare la proiect.

Jetoanele sunt afișate pe ecran în funcție de anumite criterii de sortare ce pot fi schimbată de lângă bara de căutare, de exemplu de la cel mai ieftin la cel mai scump jeton sau invers.

Deasupra secțiunii unde sunt afișate bunurile digitale există o bară de căutare pentru a căuta jetoanele după denumire, numele lor fiind format din denumirea colecției urmată de simbolul "hash", care la rândul lui este urmat de un număr ce indică numărul său din colecția respectivă.



La apăsarea unui jeton, utilizatorul este redirectionat către o pagină unde poate afla mai multe informații despre acel jeton non-fungibil, de exemplu prețul cu care poate fi cumpărat, care sunt attributele sale, cate attribute are, adresa jetonului, adresa detinătorului sau taxa care se percep pe tranzacție.

Dacă utilizatorul dorește să cumpere acel jeton, este necesar să își conexe portofelul virtual apăsând butonul “Select Wallet”.

4.3.7 Antetul paginilor

Conecțarea portofelului virtual poate fi făcută oricând din antetul pagini, acesta având o poziție fixă în partea de sus a oricărei pagini din aplicație. Tot în acest antet găsim o bară de căutare care permite utilizatorului să

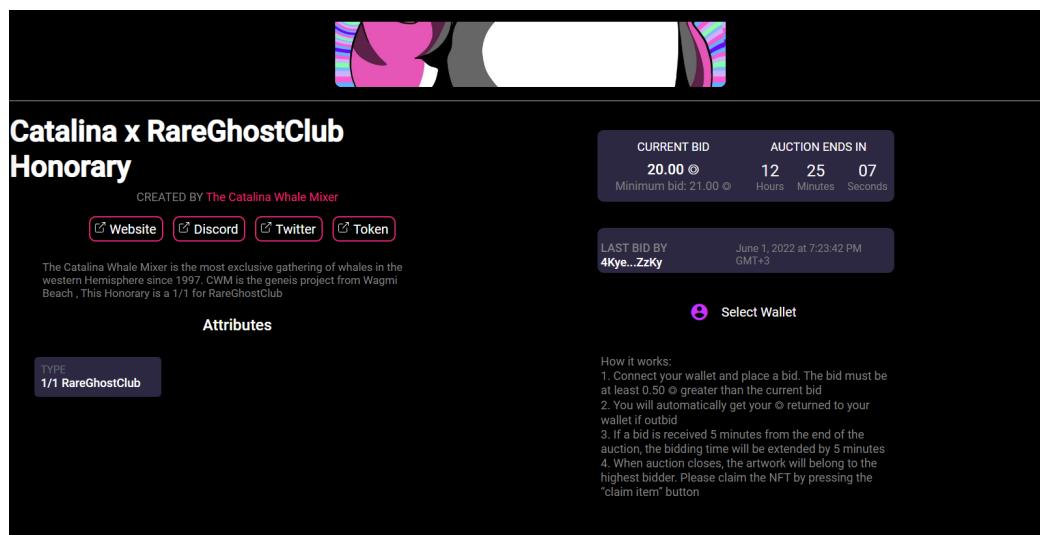
navigheze oricând către o colecție sau o licitație, nefiind nevoie să le caute în secțiunile lor specifice.



4.3.8 Pagina unde sunt afișate licitațiile

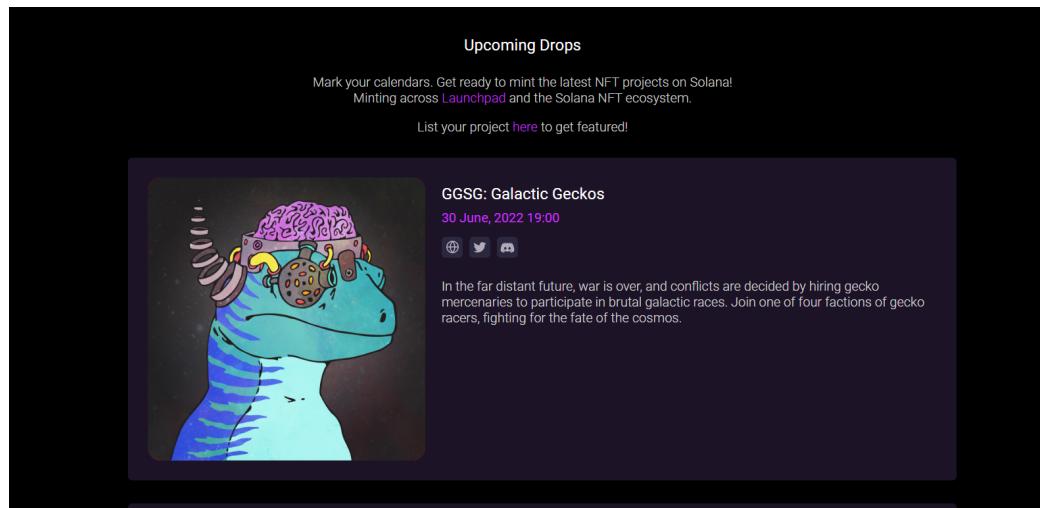
Dacă utilizatorul alege să navigheze către o licitație, acesta este redirectionat către o pagină unde poate licita pentru acel jeton, tot aici găsindu-se și informațiile referitoare la acest bun digital, cât și la colecția din care face parte, nefiind neapărat necesar de a face parte din una.

Aceste informații se referă la numărul de atribute din care este alcătuit jetonul, care sunt aceste atribute sau suma curentă sub care este pus la licitație bunul virtual.



4.3.9 Pagina unde sunt afișate informații despre proiecte în fază incipientă

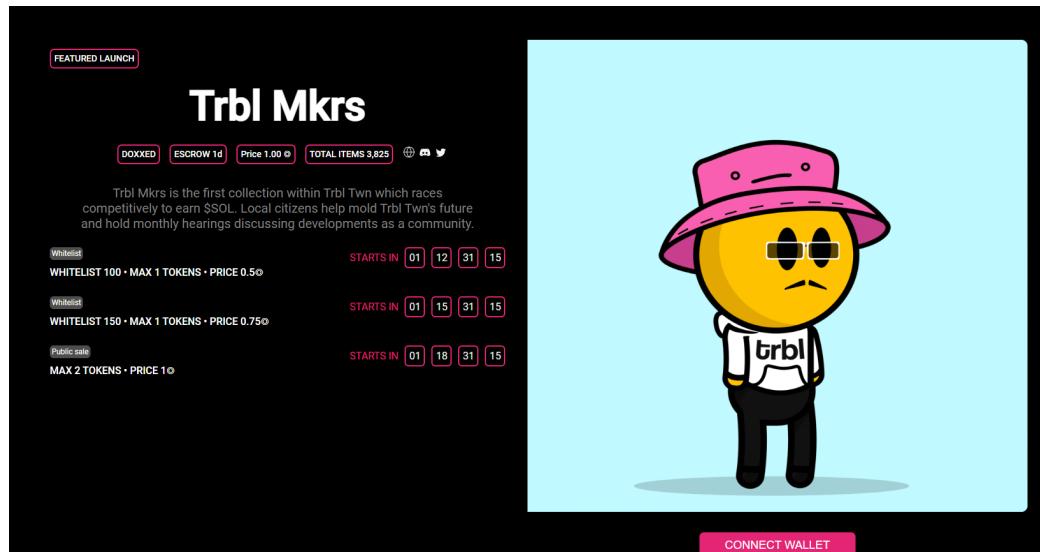
Dacă utilizatorul dorește să exploreze ce colecții vor urma să fie lansate, acesta este redirectionat către o pagină unde colecțiile sunt afișate calendaristic, în funcție de data în care vor fi lansate, tot aici găsind informațiile necesare despre viitoarele proiecte, cum ar fi link-uri către pagina de Twitter, către serverul de Discord sau către Website-ul acestora.

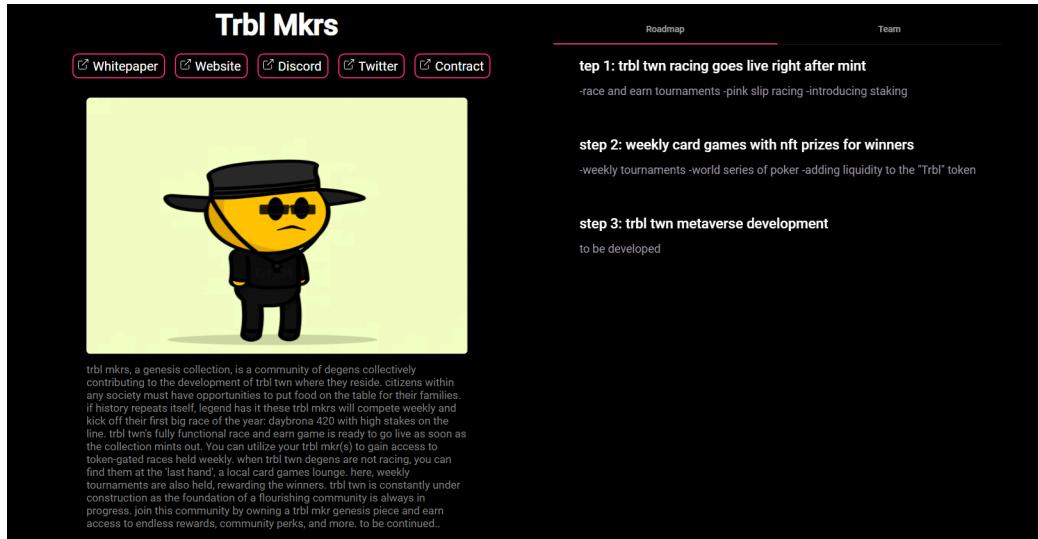


4.3.10 Pagina unde sunt afisate colectiile care vor fi lansate

În această pagină sunt prezentate informații despre colectiile existente precum momentul lansării, prețul obtinerii unui jeton nefungibil, care va fi parcursul proiectului în viitorul apropiat și ce se dorește a fi promovat sau implementat prin intermediul colecției respective.

De asemenea, pagina are o secțiune dedicată parcursului proiectului pe anumite perioade de timp și pentru dezvoltatori, unde putem găsi informații și link-uri către retelele lor sociale.





4.4 Aplicatia Java

4.4.1 Descrierea aplicatiei

4.4.2 Avantajele aplicației Java

Validarea datelor:

Preluarea informațiilor de pe partea de client în vederea prelucrării lor.

Conecțivitate:

Comunicarea cu aplicația Angular prin intermediul unei conexiuni la baza de date;

Persistență:

Metodele implementate asigură persistența obiectelor din baza de date; Aplicația Java are că scop realizarea și menținerea conexiunii cu baza de date în vederea interacționării cu aceasta.

Operațiile ce pot altera și manipula obiectele din baza de date sunt cele de tip CRUD, acesta fiind un acronim pentru Create, Read, Update, Delete. Operațiile menționate anterior sunt cele 4 operații de bază ale stocării persistente, fiind responsabile pentru adăugarea, ștergerea, modificarea și取得erea datelor din mediul de stocare. În funcție de scopul și funcționalitățile fiecărei clase, ele sunt grupate sub forma a 6 fișiere.

4.4.3 Functionalitățile aplicatiei

4.4.4 Directorul ”controllers”

Directorul ”controllers” gestionează clasele de controlori specifice cadrului Spring Boot care corespund claselor din fișierul ”data-transfer-object” responsabile de modelarea obiectelor.

Acestea au că scop gestionarea apelurilor API REST primite, construirea unui model și returnarea datelor în ”view” pentru a fi afișate drept răspuns. Adnotările @Controller sau @RestController sunt folosite pentru a adnota clasele de controlori în Spring.

Tot în aceste clase metodele sunt adnotate, în funcție de scopul metodei respective, cu adnotările specifice cadrului de lucru Spring Boot.

@GetMapping este o versiune specializată a adnotării
@RequestMapping
ce acționează că o comandă rapidă pentru
@RequestMapping(method = RequestMethod. GET).

Astfel că resursa este populată folosind metoda GET HTTP, iar adnotarea @GetMapping mapeaza cererile HTTP GET la metodele de gestionare a controlorului Spring specificate.

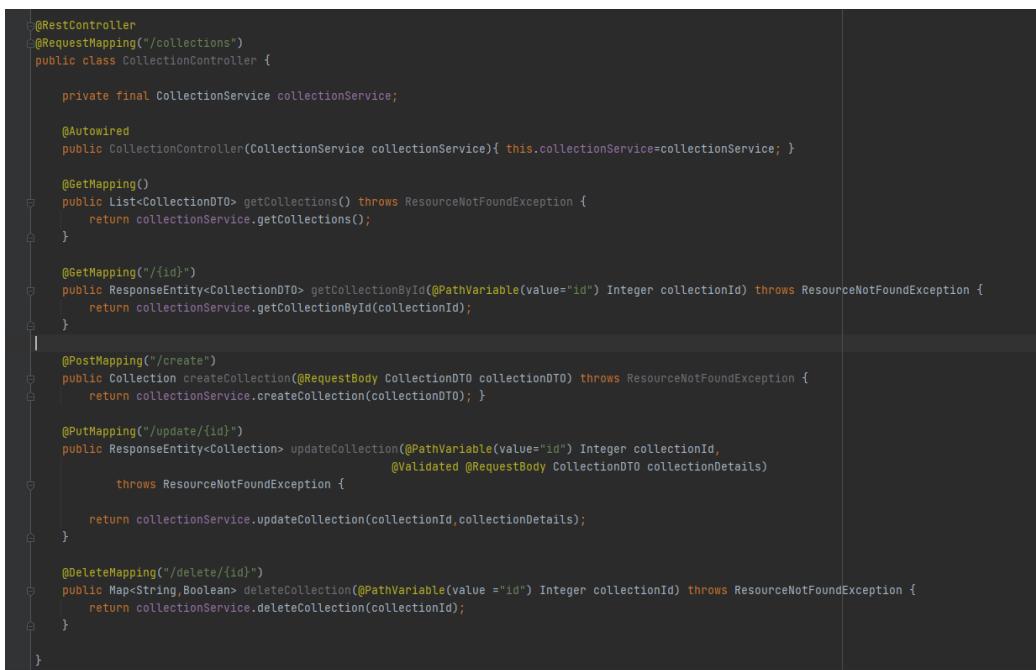
@PutMapping este o versiune specializată a adnotării
@RequestMapping
ce acționează că o comandă rapidă pentru
@RequestMapping(method = RequestMethod. PUT).

Astfel că resursa este alterată/modificată folosind metoda PUT HTTP, iar adnotarea @PutMapping mapeaza cererile HTTP PUT la metodele de gestionare a controlorului Spring specificate.

@PostMapping este o versiune specializată a adnotării
@RequestMapping
ce acționează că o comandă rapidă pentru
@RequestMapping(method = RequestMethod. POST). O nouă resursă este creată folosind metoda POST HTTP, iar adnotarea @PostMapping mapeaza cererile HTTP POST la metodele de gestionare a controlorului Spring specificate.

@DeleteMapping este o formă specializată a adnotării
 @RequestMapping
 ce funcționează că o comandă rapidă
 @RequestMapping(method = RequestMethod.DELETE)
 Resursa este ștearsă folosind metoda DELETE HTTP, iar adnotarea
 @DeleteMapping mapează cererile HTTP DELETE la metodele de gestio-
 nare a controlorului Spring specificate.

Mai jos este ilustrată o clasă de tipul controlor pentru un model ales aleator din directorul “data-transfer-object”.



```

@RestController
@RequestMapping("/collections")
public class CollectionController {

    private final CollectionService collectionService;

    @Autowired
    public CollectionController(CollectionService collectionService){ this.collectionService=collectionService; }

    @GetMapping()
    public List<CollectionDTO> getCollections() throws ResourceNotFoundException {
        return collectionService.getCollections();
    }

    @GetMapping("/{id}")
    public ResponseEntity<CollectionDTO> getCollectionById(@PathVariable(value="id") Integer collectionId) throws ResourceNotFoundException {
        return collectionService.getCollectionById(collectionId);
    }

    @PostMapping("/create")
    public Collection createCollection(@RequestBody CollectionDTO collectionDTO) throws ResourceNotFoundException {
        return collectionService.createCollection(collectionDTO);
    }

    @PutMapping("/update/{id}")
    public ResponseEntity<Collection> updateCollection(@PathVariable(value="id") Integer collectionId,
                                                       @Validated @RequestBody CollectionDTO collectionDetails)
        throws ResourceNotFoundException {
        return collectionService.updateCollection(collectionId,collectionDetails);
    }

    @DeleteMapping("/delete/{id}")
    public Map<String,Boolean> deleteCollection(@PathVariable(value = "id") Integer collectionId) throws ResourceNotFoundException {
        return collectionService.deleteCollection(collectionId);
    }
}

```

4.4.5 Directorul data-transfer-object

Fisierul “data-transfer-object” conține clase foarte asemănătoare cu cele din directorul “models” cu diferențele că proprietatea “id” lipsește din fiecare clasă și nu este folosită nici o adnotare specifică cadrului de lucru Spring Boot.

Singura adnotare pe care clasele din “data-transfer-object” o folosesc este
 @JsonAutoDetect(fieldVisibility = JsonAutoDetect.Visibility.ANY)
 fiind utilizată la nivel de clasă pentru a suprascrie vizibilitatea proprietăților
 în timpul serializării și deserializării.

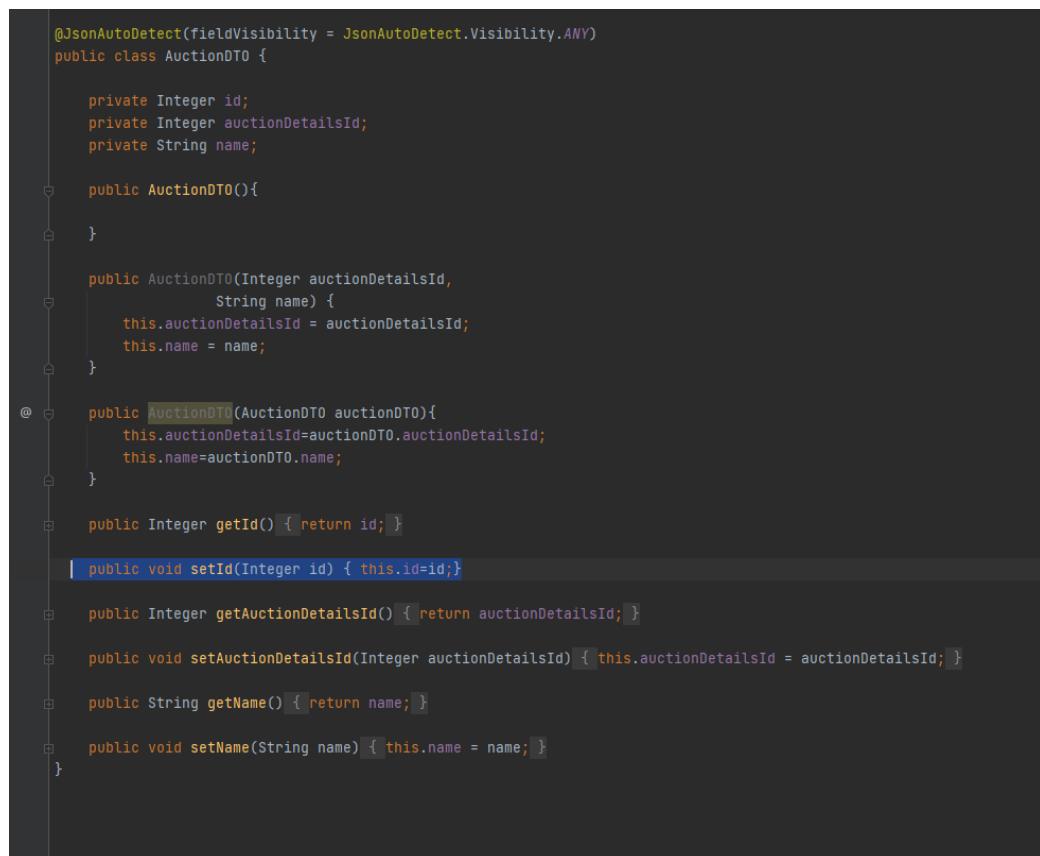
Acstea clase modelează obiectele cu care interacționează utilizatorul, nefi-
 ind nevoie de o cheie primara sau de acele adnotări deoarece ele nu interacționează

cu baza de date, acesta fiind și motivul pentru care există 2 clase de obiecte asemeneatoare.

Clasele din “models” se ocupă de interacțiunea cu baza de date, iar cele din “data-transfer-object” de interacțiunea cu utilizatorul.

Dacă de exemplu se dorescă că anumite obiecte din baza de date să fie afișate în interfață, acestea se preiau din sistemul de stocare sub forma claselor din “models”, după care există o conversie de la aceste obiecte la cele ale claselor din “data-transfer-object”, acestea într-un final fiind preluate printr-un request și afișate în interfață.

O clasă oarecare din acest director arată în felul următor:



```
@JsonAutoDetect(fieldVisibility = JsonAutoDetect.Visibility.ANY)
public class AuctionDTO {

    private Integer id;
    private Integer auctionDetailsId;
    private String name;

    public AuctionDTO(){

    }

    public AuctionDTO(Integer auctionDetailsId,
                      String name) {
        this.auctionDetailsId = auctionDetailsId;
        this.name = name;
    }

    @Override
    public AuctionDTO(AuctionDTO auctionDTO){
        this.auctionDetailsId=auctionDTO.auctionDetailsId;
        this.name=auctionDTO.name;
    }

    public Integer getId() { return id; }

    public void setId(Integer id) { this.id=id; }

    public Integer getAuctionDetailsId() { return auctionDetailsId; }

    public void setAuctionDetailsId(Integer auctionDetailsId) { this.auctionDetailsId = auctionDetailsId; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }
}
```

4.4.6 Directorul repositories

Directorul “repositories” este format din interfețe ce extind librăria JPA-Repository, aceasta constând în implementarea diferitelor funcții, metode și alte tipuri de date dependente pentru a permite persistența în aplicațiile web sau desktop concepute folosind Java.

Un Java Persistence API (JPA) este o specificație Java pentru accesarea, gestionarea și persistența datelor între obiectele Java și bazele de date relationale.

Pentru cartografierea obiect relatională, este privită drept o abordare convențională. JPA acționează că o legătură între modelele de domenii orientate pe obiect și sistemele de gestionare ale bazelor de date relationale.

Interfețele sunt adnotate cu `@Repository` care este folosită pentru a indica faptul că aceste clase sunt furnizoare de mecanisme pentru operațiile de stocare, regăsire, căutare, actualizare și ștergere pe obiecte, nefind neapărată această adnotare deoarece Spring recunoaște diferența dintre un “repository” și o clasă oarecare, ele extinzând interfața predefinită a librăriei JPARepository.

O clasă ce reprezintă un repository arată în felul următor:

```
import org.springframework.stereotype.Repository;

@Repository
public interface AuctionRepository extends JpaRepository<Auction, Integer> { }
```

4.4.7 Fisierul exceptions

Directorul “exceptions” conține clase ce implementează posibilele excepții care pot apărea atunci când baza de date este integrată sau când requesturile de tip HTTP sunt testate pentru a asigura o bună funcționare a acestora.

Astfel că sunt implementate două tipuri de excepții, “ResourceNotFoundException” fiind aruncată atunci când obiectul cu care se dorește interacțiunea nu există sau nu este găsit, iar “GlobalExceptionHandler” fiind aruncată atunci când un request HTTP nu funcționează corespunzător.

Clasa “ErrorDetails” conține detaliile erorilor care pot interveni, cum ar fi un timestamp sau mesajul generat de acea eroare.

```
public class ErrorDetails {
    private String message;
    private Date timestamp;
    private String details;

    public ErrorDetails(String message, Date timestamp, String details){
        super();
        this.message=message;
        this.timestamp=timestamp;
        this.details=details;
    }

    public String getMessage() { return message; }

    public Date getTimestamp() { return timestamp; }

    public String getDetails() { return details; }

    public void setMessage(String message) { this.message = message; }

    public void setTimestamp(Date timestamp) { this.timestamp = timestamp; }

    public void setDetails(String details) { this.details = details; }
}
```

```
import java.util.Date;

@ControllerAdvice
public class GlobalExceptionHandler {
    @ExceptionHandler(ResourceNotFoundException.class)
    public ResponseEntity<> resourceNotFoundException(ResourceNotFoundException exception, WebRequest request){
        ErrorDetails errorDetails = new ErrorDetails(exception.getMessage(), new Date(), request.getDescription( includeClientInfo: false));
        return new ResponseEntity<>(errorDetails, HttpStatus.NOT_FOUND);
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<> globalExceptionHandler(Exception exception,WebRequest request){
        ErrorDetails errorDetails = new ErrorDetails(exception.getMessage(), new Date(), request.getDescription( includeClientInfo: false));
        return new ResponseEntity<>(errorDetails, HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

```

@ResponseStatus(value= HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends Exception{
    private static final long serialVersionUID = 1L;

    public ResourceNotFoundException(String message){
        super(message);
    }
}

```

4.4.8 Fisierul services

Directorul “services” conține clase ce implementează metodele care sunt mapate în clasele din fișierul “controllers”, acestea garantând persistența datelor în aplicația web.

Pe lângă aceste implementări, clasele conțin funcții de conversie între obiectele din “models” și “data-transfer-object”.

Adnotarea @Service specifică cadrului de lucru Spring este folosită pentru a marca o clasa drept furnizor de servicii, indicând că aceasta conține doar metode ce oferă o implementare pentru anumite cerințe specifice, în cazul nostru fiind preluarea, alterarea, introducerea și ștergerea anumitor date din mediul de stocare.



4.5 Conectarea la PostgreSQL

Unul dintre avantajele folosirii cadrului de lucru Spring Boot îl reprezintă ușurința cu care se poate stabili conexiunea la diferite baze de date. Pentru a conecta o aplicație Spring Boot la o bază de date PostgreSQL sunt necesari următorii pași:

Adăugați o dependență de driver PostgreSQL JDBC, ea fiind necesară că aplicațiile Java să comunice cu un server de baze de date PostgreSQL.

Configurați proprietățile sursei de date ale informațiilor de conexiune la baza de date.

În funcție de nevoia programatorului, se adăuga o dependență pentru Spring JDBC sau Spring Data JPA, pentru instrucțiuni simple folosindu-se SpringJDBC.

Spring Data JPA este folosit pentru utilizări mai complexe, de exemplu maparea claselor Java în tabele sau a obiectelor Java în rânduri.

4.5.1 Configurarea proprietatilor

În fișierul application.properties din fișierul “resources” sunt specificate configurațiile pentru lucrul cu baza de date postgresql, portul pe care funcționează jdbc-ul furnizat de postgresql este 5432.

Configurațiile

```
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect  
spring.jpa.properties.hibernate.format-sql=true
```

sunt folosite pentru a specifica că dialectul folosit este cel al PostgreSQL, de asemenea afișând în consola mesajele de tip SQL în timpul pornirii serverului.

```
spring.datasource.url=jdbc:postgresql://localhost:5432/Licenta  
spring.datasource.driver-class-name=org.postgresql.Driver  
spring.datasource.username=postgres  
spring.datasource.password=123456  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.properties.hibernate.format_sql=true
```

4.5.2 Idei de dezvoltare a aplicatiei

Deoarece aplicația de față doar simulează tranzacțiile pe blockchain, o foarte bună îmbunătățire a aplicației este implementarea de metode prin care se realizează o conexiune a portofelului virtual la baza de date, precum și a metodelor care inițiază o tranzacție, respectiv cumpărarea, punerea spre vânzare a unui jeton sau participarea la licitații.

Această aplicație urmărește simularea tranzacțiilor de jetoane non-fungibile pentru a înțelege mai bine acest fenomen și tot ce se întâmplă pe blockchain când tranzactionăm cu acestea.

Baza de date poate fi înlocuită cu una unde sunt stocate metadatele colecțiilor, jetoanelor non-fungibile și a licitațiilor. Astfel că aplicația vă știe de unde să importe colecțiile și va permite tranzacționarea și punerea spre vanazare a acestor jetoane în interfața utilizator.

Având o bază fondată, aplicația poate fi îmbunătățită și transformată dintr-un simulator într-o platformă reală de tranzacționare a jetoanelor non-fungibile, având deja anumite funcționalități implementate precum explorarea de colecții ce vor fi lansate sau statistici ale colecțiilor existente.

Capitolul 5

Concluzia

Așadar, această aplicație poate avea o utilitate reală deoarece platformele de tranzacționat jetoane non-fungibile reprezintă atât industrie de succes de care din ce în ce mai multă lume este interesată, dar și o modalitate de promovare și susținere a proiectelor, a artiștilor grafici și a dezvoltatorilor.

Unul dintre motivele pentru care acest fenomen a devenit atât de popular îl reprezintă funcționalitățile reale pe care le au anumite jetoane non-fungibile, de exemplu ele fiind surse de venit pasiv, acțiuni, chei de acces către diferite realități virtuale, jocuri sau tehnologii de monitorizare a pieței de criptomonede și de jetoane, noduri specializate de rețea pentru efectuarea anumitor sarcini specifice, de exemplu tranzacționarea și navigarea pe rețele private ce nu numai că au o viteză mai bună, ci și numărul de conexiuni posibile la rețea este mai limitat, reducând astfel posibilele aglomerări și încetiniri ale rețelei.

Jetoanele non-fungibile reprezintă mai mult decât imagini ale deținătorilor în spațiul virtual, fiind de fapt o industrie în continuă expansiune și dezvoltare ce a schimbat viața multor persoane.

5.0.1 Bibliografie

- <https://kriptomat.io/ro/blockchain/ce-este-tehnologia-blockchain/>
- <https://ro.wikipedia.org/wiki/Blockchain>
- <https://www.oracle.com/ro/blockchain/what-is-blockchain/>
- <https://egera.com/ro/ce-este-blockchain>
- <https://crypto.ro/educatie/ce-este-blockchain-si-cum-funcioneaza/>
- <https://www.ibm.com/topics/what-is-blockchain>
- <https://www.euromoney.com/learning/blockchain-explained/what-is-blockchain>
- <https://egera.com/ro/ce-este-criptomoneda>
- <https://kriptomat.io/ro/criptomonede/tipuri-de-criptomonede/>

<https://www.nerdwallet.com/article/investing/cryptocurrency>
<https://www.coinbase.com/learn/crypto-basics/what-is-cryptocurrency>
<https://www.forbes.com/advisor/investing/cryptocurrency/what-is-cryptocurrency/>
<https://financer.com/ro/wiki/totul-despre-elrond/>
<https://biletu-zilei.com/crypto/elrond/>
<https://criptomat.io/ro/criptomonede/ce-este-nft/>
<https://angular.io/guide/what-is-angular>
<https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular>
<https://flatlogic.com/blog/what-is-angular/>
<https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/overvie>
<https://www.techtarget.com/searchapparchitecture/definition/Spring-Framework>
<https://stackify.com/what-is-spring-boot/>
<https://www.ibm.com/cloud/learn/java-spring-boot>
<https://www.postgresql.org/about/>
<https://aws.amazon.com/rds/postgresql/what-is-postgresql/>
<https://www.ibm.com/cloud/learn/java-explained>
<https://codeberryschool.com/blog/ro/notiuni-de-baza-programare-java/>
<https://ro.eferrit.com/ce-este-java/>
<https://codecool.com/ro/blog/ghid-java-incepatori/>
<https://nodejs.dev/learn>
<https://www.simplilearn.com/tutorials/maven-tutorial/what-is-maven>
<https://coinmarketcap.com/ro/>
<https://howtodoinjava.com/spring5/webmvc/controller-getmapping-postmapping/>