Razvoj pametnih ugovora

Špiranec, Leon

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Leon Špiranec

RAZVOJ PAMETNIH UGOVORA

ZAVRŠNI RAD

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Leon Špiranec
Matični broj: 16144710
Studij: Informacijski sustavi

RAZVOJ PAMETNIH UGOVORA

ZAVRŠNI RAD

Mentor/Mentorica:
Dr. sc. Marko Mijač

Varaždin, rujan 2023.

Leon Špiranec

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

_____

Sažetak

Pametni ugovori su kompjuterski programi koji mogu automatski izvršiti sve klauzule i uvjete ugovora. Kada se uvjeti pametnog ugovora između dvije stranke izvrše, automatski se izvršava isplata po naloženim uvjetima u ugovoru na transparentan način.

Ključne riječi: Pametni ugovori, Lanac blokova, Stratis plaftforma, kriptovalute

Sadržaj

1.Uvod

U suvremenom dobu, čovjek svakodnevno koristi neku vrstu usluga. Veliki broj usluga koje koristimo danas temelji se na konceptu transakcija i ugovora. Proces za provedbu ugovora i transakcije predstavlja dugotrajne i zahtjevne postupke te često dolazi do problema povjerenja da će druga strana poštivati ugovor. Tijekom povijesti, ovaj izazov je riješen uvođenjem posrednika. Kada korisnici koriste usluge posrednika, postaju ovisni o tome kako ti posrednici upotrebljavaju njihove podatke i moraju imati povjerenje u centralizirane sustave i treće strane. Postavlja se pitanje sigurnosti i povjerenja u centralizirane sustave te se traži novi pristup za veću kontrolu nad vlastitim transakcijama.

Pojavom lanca blokova i pametnih ugovora taj problem nije riješen, ali nudi mogućnost i potencijal rješavanja tog problema. No što je zapravo pametan ugovor? Riječ „pametan“ posljednjih je godina postao sve više popularan te tako postoje pametni telefoni, pametne kuće itd. Kada govorimo o pametnom ugovoru je li ispravno zaključiti da je to ugovor čija je funkcionalnost poboljšana putem softverskih aplikacija? Lanac blokova omogućuje direktnu digitalnu razmjenu vrijednosti putem kripto valuta te na isti način pametni ugovori omogućuju izravnu razmjenu između dviju strana bez mogućnosti manipulacije ili prevare jedne strane nad drugom. Pametni ugovori koriste lanac blokova tehnologiju kao temeljnu platformu za provođenje i verifikaciju uvjeta ugovora. Lanac blokova je decentralizirana, distribuirana knjiga transakcija koja osigurava sigurnost, nepovratnost i transparentnost. Koncept modernih pametnih ugovora nastao je pojavom Bitcoina, prvog kriptovalutnog sustava koji je koristio lanac blokova tehnologiju. Međutim, pravi napredak u razvoju pametnih ugovora postignut je s Ethereum platformom koja je omogućila programirane pametne ugovore putem programskog jezika Solidity.

Pametni ugovori imaju široku primjenu u raznim sektorima, uključujući financije, logistiku, nekretnine i osiguranje. Oni omogućavaju automatizaciju izvršavanja ugovora, što smanjuje potrebu za posrednicima i troškove transakcija. Također pružaju transparentnost jer se svi detalji transakcija zapisuju na lanac blokova i mogu biti javno provjereni. Ipak, pametni ugovori suočavaju se s izazovima kao što su sigurnosne ranjivosti, pravni aspekti priznavanja i provedivosti unutar tradicionalnih pravnih sustava.

Svrha ovog završnog rada je upravo detaljnije nas upoznati sa teorijskim dijelom pametnih ugovora, kako su nastali, kako funkcioniraju, gdje se koriste te detaljnije objasniti platforme programske jezike za pametne ugovore. Praktični dio ćemo objasniti implementacijom jednostavne aplikacije pametnog ugovora.

2. Metode i tehnike rada

Prilikom izrade rada koriste se sljedeći alati: **Visual Studio** kao glavno integrirano razvojno okruženje (IDE) koje nam je omogućilo pisanje, testiranje i ispravljanje grešaka u C# kodu. Testiranje i provjera s Postman-om i Swagger-om: Kako bismo osigurali ispravno funkcioniranje API-ja, koristili smo **Postman** za testiranje API endpointa. **Swagger** smo koristili za generiranje i dokumentiranje API specifikacije, olakšavajući korištenje API-ja. Integracija sa Stratis platformom: **Stratis** je poslužio kao temelj naše blockchain aplikacije, omogućavajući nam implementaciju sigurnih i decentraliziranih funkcionalnosti. Za izradu dijagrama i originalnih slika koristili smo online verziju alata **Draw.io** koji primarno služi za izradu dijagrama.

3. Pametni ugovori

3.1. Povijest pametnih ugovora

Pojam „pametni ugovor“ prvi je upotrijebio Nick Szabo, američki računalni znanstvenik u članku iz 1994. godine u kojem kaže: „Pametni ugovor je računalni protokol transakcije koji izvršava uvjete ugovora. Opći ciljevi dizajna pametnih ugovora su zadovoljiti uobičajene ugovorne, minimizirati iznimke, kako zlonamjerne tako i slučajne, te minimizirati potrebu za pouzdanim posrednicima.“

Szabova glavna teza je da su ugovori ključni za izgradnju povjerenja u društvu. Dvije godine kasnije Szabo kaže kako su ugovori osnovna građevna jedinica svakog slobodno tržišnog gospodarstva. Par godina kasnije, računalni znanstvenici i matematičari razvili su tehničke alate za automatizaciju ugovora. Još važnije, tehnologija koja omogućava rani i rudimentarni oblik pametnih ugovora već je postojala u vrijeme kada je Szabo dijelio svoje misli s javnošću. Primjer takve tehnologije je DigiCash Chaum (1988), platni sustav koji štiti privatnost korisnika. [12]

Osim toga, Szabo je vjerovao da bi pametni ugovori trebali biti provjerljivi i izvršivi kako bi imali vrijednost za društvo. Na taj način, pametni ugovori postali bi sastavni dio društva koji bi smanjio pravne barijere, smanjio troškove transakcija, skratio vrijeme izvršenja ugovora i stvorio prilike za nove oblike poslovanja. Szabo je bio u pravu u svojim predviđanjima: danas, kako se pametni ugovori razvijaju i zamjenjuju neke tradicionalne ugovore, smanjuju se troškovi i ubrzava izvršenje.

3.2. Kako funkcioniraju pametni ugovori

Pametni ugovori su računalni protokoli ili, jednostavnije rečeno, računalni kod koji se koristi za definiranje uvjeta ugovora između strana u transakciji na lancu blokova. Svi uvjeti ugovora su inkorporirani u programski kod. Na primjer, možemo definirati ugovor koji zadržava sva uplaćena sredstva dok se ne dostigne određeni cilj. Sudionici koji financiraju projekt šalju novac u pametni ugovor, a ako se dostigne potreban iznos, ugovor automatski preusmjerava ta sredstva dalje. Također je moguće definirati ugovor koji omogućuje isplatu sredstava za projekt u fazama. Ukoliko neka od faza nije ostvarena, sredstva se vraćaju sudionicima koji su financirali projekt. U stvari, sve što može biti programirano može postati dio pametnog ugovora, bez obzira na to što vam padne na pamet. U tim ugovorima se obveze sudionika izražavaju u obliku "ako-onda". Na primjer, ako strana A prenese novac,

tada strana B predaje strani A prava na kuću. Pametni ugovori mogu uključivati dva ili više sudionika, bilo da se radi o pojedincima ili organizacijama. Kada se zadovolje postavljeni uvjeti, pametni ugovori automatski izvršavaju transakciju i osiguravaju poštivanje dogovora. Prema [20]

Pametni ugovori omogućuju razmjenu novcem, dobrima, nekretninama, vrijednosnim papirima i drugim sredstvima. Ti ugovori se pohranjuju i repliciraju u distribuiranoj knjizi, gdje informacije ne mogu biti krivotvorene ili izbrisane. Istovremeno, šifriranje podataka osigurava anonimnost strana ugovora. Važna karakteristika pametnih ugovora je da mogu djelovati samo unutar svog digitalnog sustava na sredstvima koja su mu dostupna.

3.3 Sigurnost pametnih ugovora

Kada sve ide prema planu, pametni ugovori ubrzavaju usvajanje i rast lanca blokova, no često se pojavljuju sigurnosne slabosti koje dovode do gubitka povjerenja i krađe sredstava. Sigurnost pametnih ugovora odnosi se na načela i prakse koje programeri, korisnici i razmjene koriste prilikom stvaranja ili interakcije s pametnim ugovorima. Lanac blokova je dinamična industrija vrijedna milijarde dolara, stoga zlonamjerni akteri često traže ranjivosti u pametnim ugovorima kako bi brzo profitirali. Ove slabosti mogu rezultirati pražnjenjem svih sredstava iz novčanika pametnog ugovora. Kao što smo prije naveli, pametni ugovori djeluju unutar okvira nepromjenjive decentralizirane lanac blokova mreže, što znači da njihovi rezultati ne mogu biti krivotvoreni u svrhu ilegalne dobiti. Međutim, ta nepromjenjivost nije samo prednost, već i nedostatak. Na primjer, 2016. godine cyber-kriminalci su izvršili hakerski napad na decentraliziranu autonomnu organizaciju „The DAO“ i nezakonito prisvojili milijune dolara u Ethereumu iskorištavajući ranjivosti u kodu pametnog ugovora. Budući da je pametni ugovor „The DAO“ bio nepromjenjiv, razvojni programeri nisu mogli ispraviti kod. Općenito se vjeruje da je izuzetno teško izvršiti hakerski napad na dobro napisane pametne ugovore te da oni predstavljaju najpouzdaniji način pohrane dokumenata u digitalnom svijetu. Međutim, važno je imati na umu da svaki kod pišu ljudski programeri koji su podložni greškama. Budući da je pametni ugovor vidljiv svim korisnicima lanaca blokova, moguće ranjivosti također postaju vidljive u cijeloj mreži i nije uvijek moguće eliminirati ih zbog nepromjenjivosti.

Neke od poznatih grešaka prouzrokovane ljudskim faktorom su sljedeće:

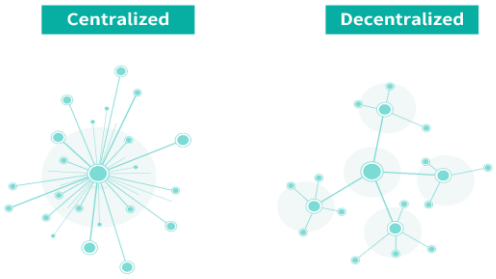
- **Rekurzivni pozivi**: Pametni ugovor poziva drugi vanjski ugovor prije potvrde promjena. Međutim, nakon toga, vanjski ugovor može neovlašteno ponovno pozvati početni pametni ugovor jer njegovo stanje još nije ažurirano.

- **Prekoračenje**: Prilikom izvršavanja aritmetičkih izračuna, pametni ugovor može premašiti ograničenje pohrane. To može rezultirati neispravnim izračunom iznosa.
- **Iskorištavanje prije vremena**: Loše dizajniran kod sadrži informacije o nadolazećim transakcijama koje vanjske strane mogu iskoristiti za vlastitu korist.

Sigurnost pametnih ugovora može se poboljšati kroz temeljan pregled koda, testiranje i reviziju, primjenu najboljih praksi, formalnu verifikaciju, kontinuirano praćenje te pravovremene nadogradnje sigurnosnih mjera. Međutim, važno je imati na umu da nijedan sustav nije potpuno neprobojan te je redovito održavanje i praćenje najnovijih sigurnosnih praksi ključno za osiguranje sigurnosti pametnih ugovora.

3.4 Decentralizirani sustav

Kao što smo ranije spomenuli, pametni ugovori su programirani ugovori koji se izvršavaju na lancu blokova. Oni iskorištavaju decentralizirani sustav i tehnologiju lanac blokova kako bi automatizirano provodili uvjete ugovora. Decentralizirani sustavi igraju ključnu ulogu u kontekstu pametnih ugovora. Umjesto da se oslanjaju na jedan centralni autoritet, pametni ugovori koriste distribuirani sustav s više čvorova koji provjeravaju i izvršavaju ugovore neovisno. Decentralizacija donosi niz prednosti za pametne ugovore. To osigurava transparentnost, jer izvršavanje ugovora i rezultati su vidljivi svim sudionicima mreže. Također pruža sigurnost jer se ne oslanja na jedan centralni autoritet koji bi mogao biti podložan napadima ili manipulacijama. Osim toga, decentralizacija gradi povjerenje jer izvršavanje ugovora provjeravaju više neovisnih čvorova. Decentralizacija također pruža otpornost na kvarove i toleranciju na greške. Ako neki čvorovi ne uspiju ili izađu iz mreže, ostatak sustava i dalje može nastaviti s izvršavanjem ugovora. Ovo osigurava pouzdanost i dostupnost pametnih ugovora.



Slika 1: Prikaz centraliziranog i decentraliziranog sustava (Izvor:[10])

3.5 Lanac blokova tehnologija

Najjednostavnije rečeno, lanac blokova je digitalna datoteka koja se koristi za pohranu podataka, a ti podaci su distribuirani (duplicirani) na mnogo računala (decentraliziran).

U detaljnijem i tehničkom smislu, lanac blokova se temelji na peer-to-peer[1] tehnologiji, gdje cijeli sustav čine čvorovi (eng. *peers*). Svaki čvor u mreži odobrava i bilježi svaku transakciju koja se događa u mreži. To znači da se svaka nova transakcija koja se dodaje u mrežu provjerava i dodaje u knjigu svakog čvora. Ovaj decentralizirani način bilježenja transakcija u knjizi naziva se tehnologija distribuirane knjige (eng. DLT).

Lanac blokova je vrsta DLT-a. Svaka pohranjena transakcija u lancu blokova naziva se blokom. Svaki blok je povezan s prethodnim blokom korištenjem kriptografskog sažetka. Dakle, ako netko želi manipulirati određenim blokom, morao bi manipulirati sa svim prethodnim blokovima u lancu, što je nemoguće/iznimno teško jer svi čvorovi imaju istu knjigu i svaka promjena je uočljiva. Time se stvara nepromjenjiv zapis blokova koji ne zahtijeva vanjski autoritet. Dakle, u najjednostavnijem obliku, lanac blokova je lanac blokova. To je proces ili tehnologija zapisivanja informacija u obliku blokova koji sprječava promjene, hakiranje ili prevaru. (Dugan, 2018, str. 35).

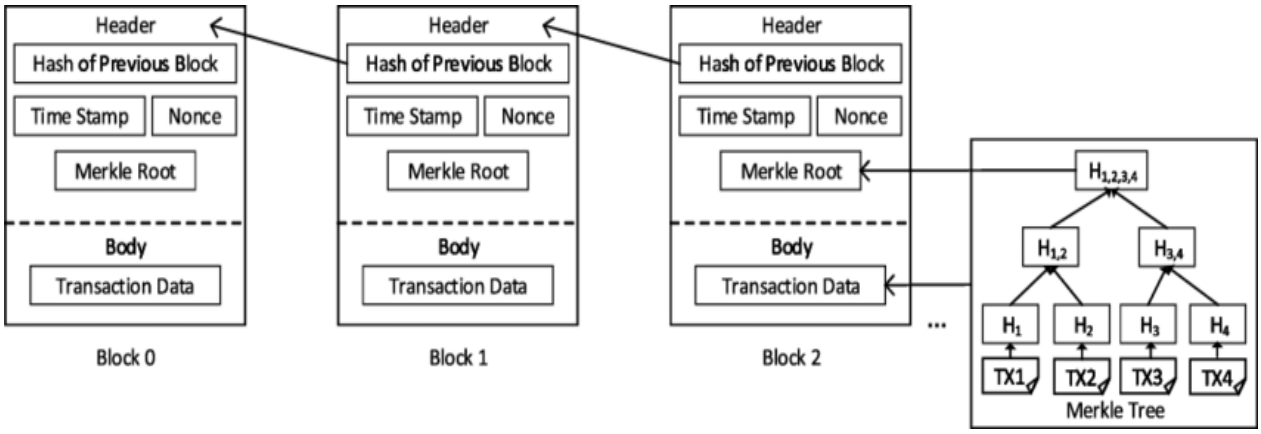
3.5.1. Struktura lanca blokova

Da bismo što jasnije shvatili strukturu lanca blokova, proći ćemo prvo kroz osnovne komponente potrebne za strukturu lanca blokova. Prema Duganu Osnovne komponente arhitekture lanca blokova su čvor, transakcija, lanac, rudari, konsenzus i blok (Dugan, 2018, str. 40).

- **Čvor** - korisnik ili računalo unutar lanac blokova arhitekture (svaki ima neovisnu kopiju cijelog lanac blokova zapisa)
- **Transakcija** – najmanja jedinica sustava (zapisi, informacije itd.) koja je svrha lancu blokova.
- **Lanac** - slijed blokova u određenom redoslijedu
- **Rudari** - određeni čvorovi koji obavljaju proces verifikacije bloka prije dodavanja bilo čega u strukturu lanca blokova

[1] Peer-to-peer - mrežni model u kojem računala ili hardverski uređaji razmjenjuju datoteke. Neki stručnjaci opisuju ga kao sustav "ravnopravnog klijenta", gdje umjesto pristupa datotekama s poslužitelja, "peer" računala jednostavno ih međusobno izmjenjuju.

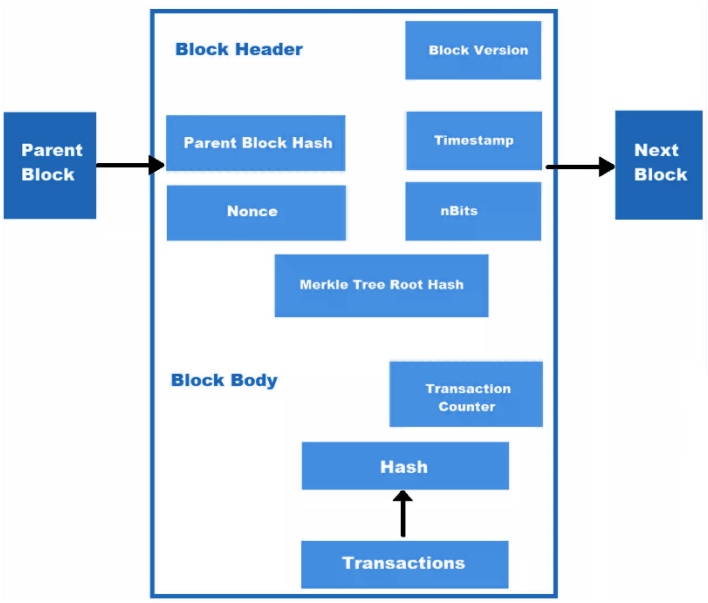
- **Konsenzus** - skup pravila i dogovora za provođenje operacija u lancu blokova
- **Blok**: struktura podataka koja se koristi za pohranu skupa transakcija koje se distribuiraju svim čvorovima u mreži. Svaki dan se događa veliki broj transakcija diljem svijeta. Važno je da korisnici prate te transakcije, što čine uz pomoć blokovske strukture. Samu strukturu lanca blokova i bloka objasnit ćemo u nastavku. Kao što smo prije napomenuli, lanac blokova je linearan lanac blokova, a jedan blok sadrži skup transakcija i druge bitne pojedinosti koje ćemo u nastavku objasniti. Svi blokovi su linearno povezani te kriptografsko osigurani.



Slika 2: Struktura lanca blokova (izvor [8])

Tako su svi blokovi međusobno linearno povezani preko hash[2] vrijednosti prethodnog bloka. Prethodni hash bloka se koristi za izračunavanje hash vrijednosti trenutnog bloka dok prvi blok nema prethodni hash bloka te se on naziva "Genesis blok". Kod dodavanja novog bloka u mrežu, lanac blokova koristi mehanizme konsenzusa kao što su dokaz o radu (eng. *proof of work-PoW*) te dokaz o udjelu (eng. *proof of stake-PoS*) koje ćemo u nastavku detaljnije objasniti. Svaki blok sadrži različita polja koja se mogu grubo kategorizirati kao na slici ispod.

[2] Hash - oznaka kojom se označuju i povezuju blokovi



Slika 3: Struktura jednog bloka (izvor: vlastita izrada)

Gornja slika će stvoriti jednostavniju vizualizaciju konceptualnog bloka u vašoj glavi. Međutim, stvarni blok sadrži puno više informacija od gornje slike. Sada ćemo objasniti značajnije elemente bloka koji su prikazani na slici:

- **Veličina bloka:** Veličina ovog polja je 4 bajta i sadrži veličinu bloka.
- **Zaglavlje bloka:** Veličina zaglavlja bloka je 80 bajta. Dodatno sadrži različita polja.
- **Brojač transakcija:** Ovo polje sadrži broj transakcija, a njegova veličina je između 1 i 9 bajta.
- **Transakcije:** Ovo polje sadrži transakcije bloka i njegova veličina je varijabilna.

Zaglavlje bloka sadrži različita polja meta podataka, a to su:

- **Hash prethodnog bloka**: Svako zaglavlje bloka daje informacije o prethodnom ili roditeljskom bloku. Ovo polje sadrži hash vrijednost prethodnog bloka i ta referenca povezuje sve blokove. Veličina ovog polja je 32 bajta.
- **Version**: Ovo polje pohranjuje broj verzije kako bi se prikazale nadogradnje softvera. Veličina polja verzije je 4 bajta.
- **nBits**: Polje od 4 bajta koje govori o složenosti dodavanja bloka. Poznato je i kao "Difficulty". Prema PoW-u, hash bloka trebao bi biti manji nBits.
- **Timestamp**: Ovo polje sadrži vrijeme kada je blok stvoren. Veličina ovog polja je 4 bajta.
- **Nonce**: Vrijednost koja se koristi tijekom rudarenja bloka. Veličina ovog polja je također 4 bajta.

- **Merkle tree root**: Merkle stablo je struktura koja se dobiva hashiranjem transakcijskih podataka bloka. Korijen ovog stabla pohranjen je u zaglavlju bloka pod ovim poljem. Veličina polja je 32 bajta.

Svaki novi zapis ili transakcija unutar lanca blokova implicira izgradnju novog bloka. Svaki zapis se zatim provjerava i digitalno potpisuje kako bi se osigurala njegova autentičnost. Prije nego što se taj blok doda u mrežu, mora ga provjeriti većina čvorova u sustavu.

3.5.2. Osnovni tipovi lanca blokova

Kada govorimo o vrstama lanac blokova tehnologije, razlikujemo 4 osnovnih tipova: javni, privatni, konzorcijski i hibridni lanac blokova[3] (Bashir, 2017, str. 545). Javni lanci blokova dozvoljavaju svakome da se pridruži i potpuno su decentralizirani. Na javnim lancima blokova, svi čvorovi imaju jednaka prava pristupa lancu blokova, mogu stvarati nove blokove podataka i potvrđivati blokove podataka. Do sada se javni lanci blokova uglavnom koriste za razmjenu i rudarenje kriptovaluta. Na tim javnim lancima blokova, čvorovi "rudare" kriptovalute stvarajući blokove za transakcije koje su zatražene na mreži rješavanjem kriptografskih jednadžbi. Kao nagradu za svoj rad, rudari dobivaju određenu količinu kriptovalute.

Privatni lanci blokova su pod kontrolom jedne organizacije. U privatnom lancu blokova, jedna centralna vlast određuje tko može biti čvor i ne mora dodijeliti jednaka prava svim čvorovima za obavljanje funkcija. Privatni lanci blokova su samo djelomično decentralizirani jer je javni pristup tim lancima blokova ograničen. Primjeri privatnih lanca blokova su Ripple i Hyperledger.

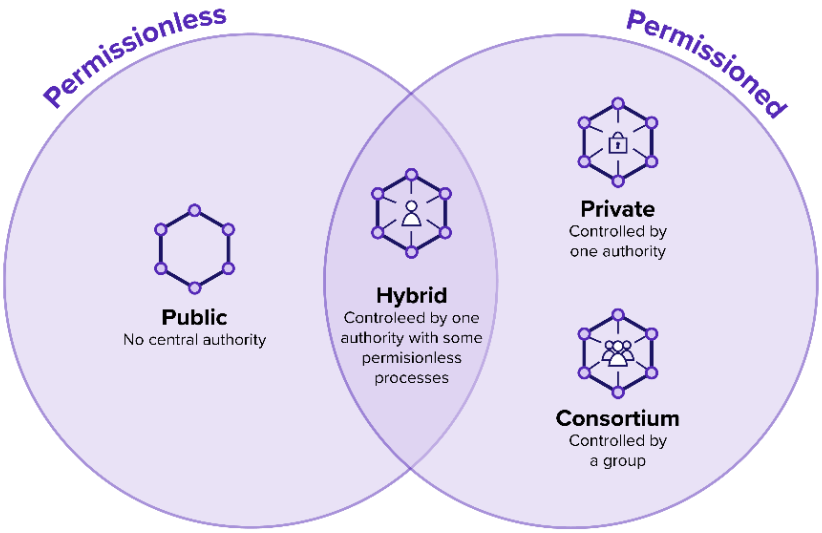
I privatni i javni lanci blokova imaju svoje nedostatke - javni lanci blokova često imaju duža vremena potvrde novih podataka u usporedbi s privatnim lancem blokova, dok su privatni lanci blokova osjetljiviji na prijevare i neželjene aktere. Kako bi se riješili ovi nedostaci, razvijeni su konzorcijalni i hibridni lanci blokova.

Hibridni lanac blokova je kombinacija privatnog i javnog lanca blokova, kako i sam naziv sugerira. Ova vrsta nastoji iskoristiti prednosti obje vrste. U hibridnom lancu blokova, nemaju svi članovi jednaka prava za potvrdu transakcija. Samo određeni broj članova ima određene privilegije vezane uz valjanost transakcija. Ostali članovi mogu sudjelovati u potvrdi, ali taj odabrani broj članova mora postići konsenzus prije provedbe transakcije.

[3] Lanac blokova (ili Blockchain ne engleskom) - je tehnologija koja se može opisati kao digitalno knjigovodstvo. Lanac blokova možete zamisliti kao “digitalnu knjigu” koja sadrži sve bitne informacije o transakcijama

Ovakav pristup pruža djelomičnu decentralizaciju ovom tipu lanca blokova, iako još uvijek naginje prema centralizaciji, što rezultira bržim izvršavanjem transakcija na mreži. Glavne karakteristike hibridnog lanca blokova su da čvorovi mogu biti samo prethodno odabrani članovi, da nije svima dopušten pristup mreži te da su identiteti članova na mreži unaprijed poznati.

Četvrti i posljednji, konzorcijalni lanac blokova, također poznat kao federirani lanac blokova, sličan je hibridnom lancu blokova jer ima značajke privatnog i javnog lanac blokova. No, razlikuje se po tome što se više organizacijskih članova surađuje na decentraliziranoj mreži. U osnovi, konzorcijalni lanac blokova je privatni lanac blokova s ograničenim pristupom određenoj grupi, čime se eliminiraju rizici koji proizlaze iz toga da samo jedan entitet kontrolira mrežu privatnog lanca blokova. U konzorcijalnom lancu blokova, postupci konsenzusa kontroliraju se unaprijed definiranim čvorovima. Postoji čvor za provjeru ispravnosti koji inicira, prima i potvrđuje transakcije. Članovi čvorova mogu primati ili inicirati transakcije.



Slika 4: Osnovni tipovi lanaca blokova (izvor: [16])

3.5.3. Kako funkcionira lanac blokova

S obzirom da smo u prijašnjim odlomcima već detaljnije objasnili struktura i osnovne komponente bloka, valja pojasniti 4 ključne komponente kako bi u potpunosti shvatiti kako funkcionira lanac blokova, a to su distribuirana knjiga (eng. *The Distributed Ledger*), mreža ravnopravnih članova (eng. *peer-to-peer-P2P*), Konsenzus mehanizam i kriptografski ključevi. Prema [17]

Kriptografski ključevi su podijeljeni na privatne i javne ključeve, a svaki sudionik ili čvor posjeduje oba ključa. Koriste se za stvaranje digitalnog potpisa. Digitalni potpis predstavlja jedinstvenu i sigurnu digitalnu identifikaciju te je najvažniji aspekt lanac blokova tehnologije. Svaka transakcija je ovjerena digitalnim potpisom vlasnika.

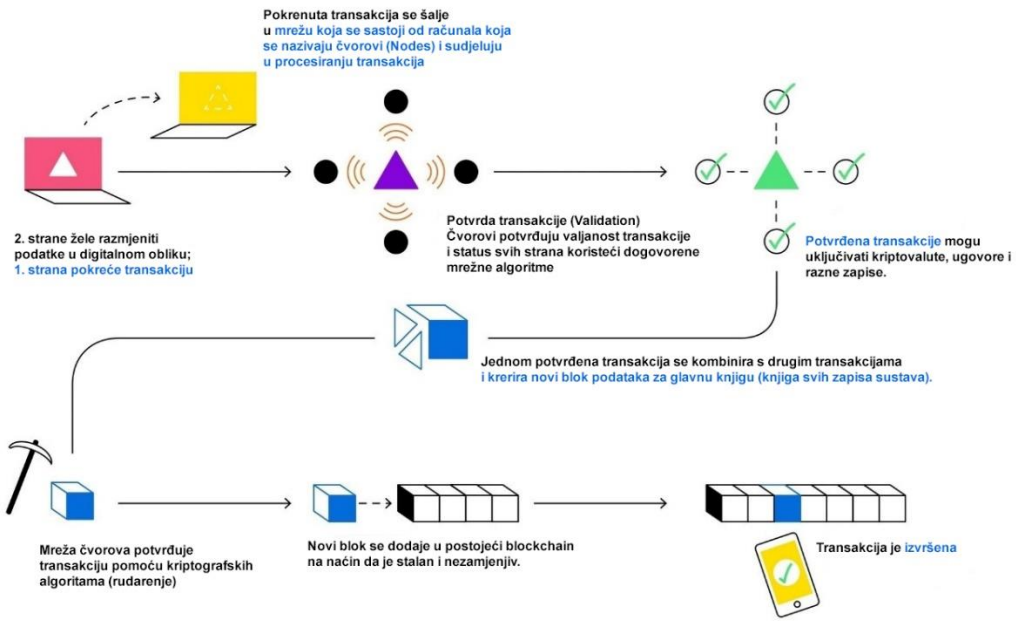
Svaki novi korisnik (čvor) koji se pridružuje **peer-to-peer mreži** lancu blokova prima punu kopiju sustava. Kada se stvori novi blok, on se šalje svakom čvoru unutar lanac blokova sustava. Svaki čvor može neovisno provjeravati je li navedena informacija ispravna. Ako je sve u redu, blok se dodaje lokalnom lancu blokova u svakom čvoru. Moramo napomenuti da to radi prvenstveno na decentraliziranom sustavu.

Konsenzusni mehanizmi skup su pravila koja određuju kako decentralizirana računalna mreža postiže dogovor o tome koje su transakcije valjane, a koje nisu. To je apsolutno neophodna komponenta svake lanac blokova mreže. Dva najpopularnija mehanizma su dokaz o udjelu (eng. *Proof-of-stake*) i dokaz o radu (eng. Proof-of-work) mehanizam. U Proof of Work lancu blokova, svi čvorovi sudjeluju u procesu rudarenja kriptovaluta kako bi osigurali točnost svakog novo izrađenog bloka. U Proof of Stake lancu blokovima, svi sudionici verificiraju sve transakcije, uključujući i transakcije uključene u blokove koje su drugi čvorovi stavili.

Sve transakcije su pohranjene u digitalnoj knjizi koja se naziva **digitalna knjiga (DLT)**. digitalna knjiga funkcionira poput tablice koja sadrži sve brojne čvorove u mreži i ima povijest svih kupnji koje je svaki čvor izvršio. Informacije sadržane u digitalnoj knjizi su izuzetno sigurne, a digitalni potpis ih štiti od manipulacija. Ono što je posebno kod ove knjige je da svi mogu vidjeti podatke, ali nitko ih ne može narušiti.

Sada kada smo ukratko objasnili najvažnije komponente za funkcioniranje lanca blokova, proći ćemo koracima za dodavanje bloka u lanac blokova mrežu:

1. Čvor stvara novu lanac blokova transakciju koristeći svoju računalnu snagu.
2. Transakcija se dijeli sa svim čvorovima na lanac blokova platformi.
3. Svi čvorovi računaju jednadžbe, provjeravaju valjanost lanac blokova transakcije i mogu koristiti algoritme konsenzusa za donošenje odluke.
4. Čvorovi jednoglasno glasuju za uključivanje transakcije u blok.
5. Jedan blok može sadržavati različite transakcije sve dok se ne napuni.
6. Na blok se primjenjuje jedinstveni identifikator koji sadrži kriptografski sažetak trenutnog bloka i sažetak prethodnog bloka.
7. Blok se dodaje u lanac, a knjiga se ažurira na svim čvorovima.
8. Proces dodavanja blokova u mrežu lanca blokova naziva se rudarenje.



Slika 5: Kako funkcionira lanac blokova tehnologija (Izvor: vlastita izrada)

4. Platforme za pametne ugovore

Pametni ugovori omogućuju korisnicima pisanje izvršivog koda koji dekodira poslovnu logiku i izvršava se na lancu blokova ili drugom tipu decentraliziranog registra. Ovisno o vašim ciljevima, postoji mnogo načina za korištenje platformi za pametne ugovore kako biste ove moćne aplikacije implementirali u stvarni rad.

Prema Sanjivu Maewallu [15], platforme za pametne ugovore su decentralizirani sustavi koji omogućuju samoizvršavanje ugovora na lancu blokova. Ove platforme podržavaju sigurno, transparentno i neovlašteno izvršavanje unaprijed definiranih uvjeta. Postoji desetine platformi za pametne ugovore, a područje se brzo razvija. Umjesto da obuhvatimo sve njih, korisnije je usredotočiti se na glavne kategorije i najveće ili najzanimljivije sudionike. Imajući to na umu, evo nekoliko vodećih platformi za pametne ugovore u ključnim kategorijama.

4.1 Ethereum

Ethereum predstavlja decentraliziranu platformu koja se smatra najmoćnijom mrežom za izradu pametnih ugovora i decentraliziranih aplikacija. Ethereum je stvoren krajem 2013. godine od strane Vitalika Buterina, programera i istraživača u području kriptovaluta. Podržava Solidity, varijantu programskog jezika JavaScript koja je odmah bila dostupna razvojnim programerima i brzo je privukla veliku zajednicu programera i financijskih podržavatelja. Ethereum je također bio pionir u stvaranju EVM-a, koji je postao standard za druge platforme za pametne ugovore te kojeg ćemo kasnije u nastavku detaljnije objasniti. Tržišna kapitalizacija Ethereuma značajno nadmašuje konkurenciju. Sredinom 2023. godine dosegnula je 223 milijarde dolara, što je višestruko više od ukupnog kapitala svih ostalih platformi za pametne ugovore zajedno. Glavna prednost velike tržišne kapitalizacije je ta što uspješne aplikacije pametnih ugovora imaju potencijal bržeg rasta u puno većoj virtualnoj ekonomiji. Ethereum je prvi izbor za programere koji žele sudjelovati u najvećem ekosustavu pametnih ugovora s novom i sjajnom idejom. Međutim, neki promatrači ističu da performanse Ethereuma nisu tako brze kao kod nekih novih platformi. Također, greške u pametnim ugovorima mogu biti skupe i teške za ispraviti.

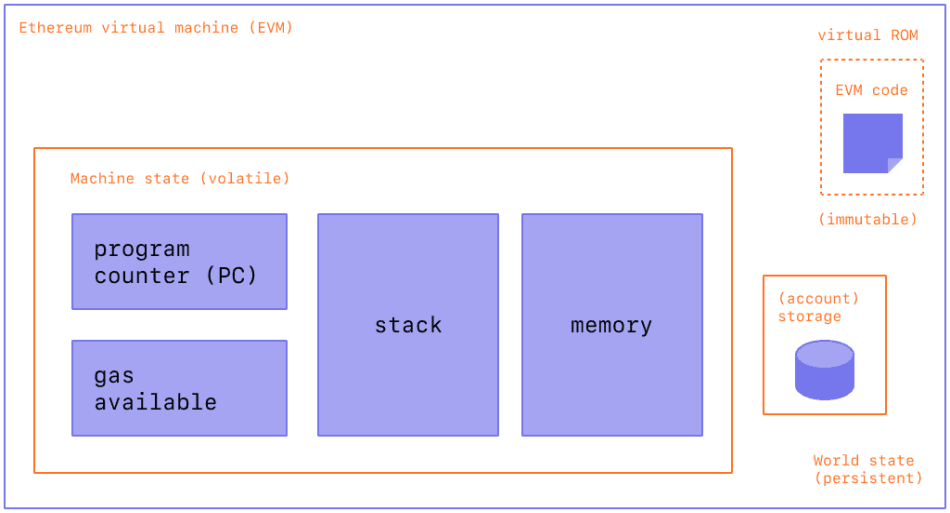
4.1.1 Ethereum virtualna mašina (EVM)

Ethereum virtualna mašina (eng. *Ethereum Virtual Machine-EVM*) je posebna virtualna mašina koja neprekidno radi i čije nepromjenjive operacije određuju stanje svakog

bloka u Ethereum lancu blokova. EVM ne samo da upravlja time što čvorovi mogu ili ne mogu raditi na distribuiranoj knjizi koju održava Ethereum lanac blokova, već također definira specifična pravila za promjenu stanja iz bloka u blok. Sposobnost za potonje omogućuje funkcionalnost pametnih ugovora za koju je Ethereum postao poznat.

Da bismo shvatili što Ethereum Virtual Machine radi, trebamo sagledati svaku od različitih funkcija koje obavlja kako bi osigurala glatko funkcioniranje Ethereum mreže. Za svaki ulaz koji primi, EVM proizvodi izlaz koji je deterministički po prirodi i slijedi matematičku funkciju u najjednostavnijem smislu.

Djelujući kao stroj sa stogom koji gura prolazne vrijednosti prema dolje i prema gore na stogu, EVM ima dubinu od 1024 stavke, pri čemu je svaka od njih 256-bitna riječ. Također održava privremenu memoriju u obliku niza bajtova, koja se mijenja između dvije transakcije na Ethereum lancu blokova. Pametni ugovori čiji je kod preveden izvršavaju se od strane EVM u obliku skupa od 140 standardnih operativnih kodova, dok su i druge operacije stoga specifične za lanac blokova također implementirane od strane njega.



Slika 6. Shematski prikaz Ethereum virtualne mašine (izvor:[1])

No koji je zapravo cilj Ethereum virtualne mašine? Etherem virtualna mašina pouzdano pokreće sve aplikacije koje se izvode na Ethereum mreži, bez ikakvih većih izvještenih prekida rada. Za razvojne programere, EVM djeluje kao sveobuhvatni program koji pokreće manje izvršive programe poznate kao pametni ugovori u Ethereumu, pružajući im slobodu da ove pametne ugovore pišu u različitim programskim jezicima, uključujući Solidity kojeg ćemo u nastavku detaljnije objasniti. Za kraj, može se reći da je EVM umetnut unutar svakog Ethereum čvora kako bi izvršavao pametne ugovore koristeći bajt kod umjesto osnovnog

programskog jezika, čime se izolira fizičko računalo domaćina od strojnog koda na kojem Ethereum radi.

4.1.2 Solidity

Solidity je objektno orijentirani programski jezik visoke razine stvoren specifično od strane tima Ethereum mreže za izgradnju i dizajniranje pametnih ugovora na lanac blokova platformama. Inspiriran je jezicima C++, Python i JavaScript.

Solidity je statički tipiziran, koji pruža podršku za nasljeđivanje, knjižnice i složene korisnički definirane tipove. Pomoću Solidityja moguće je kreirati ugovore za primjene kao što su masovno financiranje (eng. *Crowdfunding*), slijepi natječaji i višenamjenski novčanici. Solidity nije težak za naučiti. Međutim, pametni ugovori u Solidityju grade se na lancu blokova, pa ih nije moguće mijenjati ili brisati. To ih čini vrlo sigurnim načinom za prijenos bilo čega od vrijednosti, ali isto tako može predstavljati nedostatak za programera. Kod iza pametnih ugovora u Solidityju mora biti besprijekoran i bez grešaka jer ne možete ispraviti pogreške ili poništiti transakcije. Ažuriranje Ethereum koda je jedini način za ispravak ranjivosti, što nije jednostavan zadatak.

4.2 Cardano

Cardano je predstavljen kao energetski učinkovitija alternativa Ethereumu 2015. godine od strane Charlesa Hoskinsona, koji je također bio suosnivač Ethereuma. 2021. godine je predstavio svoju Plutus značajku za izradu pametnih ugovora. Cardano je novija platforma s manje funkcionalnosti u usporedbi s drugima na ovom popisu. Međutim, ima drugu najveću tržišnu kapitalizaciju od otprilike 13 milijardi dolara, čineći je zanimljivim izborom za programere koji žele iskoristiti veliku token ekonomiju izvan Ethereuma.

Slika 7. Cardano logo (izvor [3])

4.2 Solana

Sredinom 2023. godine, Solana lanac blokova vrijedan je oko 8 milijardi dolara, što ga čini manjim u obujmu. Međutim, stručnjaci tvrde da je njegova učinkovitost nadmašila onu

Ethereuma i nekoliko drugih javnih lancu blokova. Trenutno Solana obrađuje između 5.000 do 10.000 transakcija u sekundi, a potencijalno može narasti na stotine tisuća. Ključni čimbenici koji doprinose toj brzini uključuju podršku za virtualnu mašinu za implementaciju pametnih ugovora, kao i programskih jezika C i Rust za njihovo pisanje. Ta kombinacija povoljno pozicionira Solanu za razvojne programere koji traže spajanje visokih brzina transakcija s ekonomskim izgledima koje pruža javni lanac blokova.

Slika 8. Solana logo (izvor [4])

4.3 Hyperledger Fabric

Sljedeća platforma je privatni tip lancu blokova, odnosno samo će vlasnik imati pravo da napravi promjene koje treba napraviti. Takav primjer je platforma Hyperledger Fabric.

IBM je razvio Hyperledger Fabric, a zatim ga predao Linux Foundationu kako bi stvorili lanac blokova platformu za poslovne svrhe s modularnom arhitekturom i različitim mehanizmima konsenzusa. Omogućuje preciznu kontrolu vidljivosti podataka i povjerljivosti tako da samo ovlaštene osobe mogu pristupiti podacima. Također podržava više programskih jezika, uključujući JavaScript, Go i Java, te je dizajniran za rukovanje velikim obimom transakcija.

Slika 9. Hyperledger fabric logo (izvor [5])

4.4 Stratis

Stratis Platforma je snažna i prilagodljiva platforma za razvoj lanca blokova koja je prvobitno dizajnirana za potrebe stvarnih poslovnih subjekata, posebno u sektoru financijskih usluga. Nudi sveobuhvatan skup alata i usluga koji omogućuju programerima i tvrtkama učinkovito stvaranje, testiranje i implementaciju aplikacija temeljenih na lancu blokova. Jedna od njezinih istaknutih značajki je kompatibilnost s više programskih jezika, uključujući

C#, Java i JavaScript, čime je dostupna širem krugu programera. Stratis pruža širok spektar usluga izvan same tehnologije. Nudi savjetodavne usluge kako bi pomogao tvrtkama u određivanju najboljih primjena i strategija implementacije lanac blokova tehnologije. Stratis Platforma je započela kao konzultantska tvrtka sa sjedištem u Londonu, ali je proširila svoje djelovanje na međunarodnoj razini kako je potražnja za njenim uslugama rasla.

Stratis full node - Puni čvor ima ključnu ulogu u očuvanju sigurnosti i decentralizacije Stratis lanac blokova mreže jer svaki čvor provjerava i potvrđuje transakcije te čuva potpunu povijest blokova. Također korisnicima omogućuje provjeru statusa svojih transakcija i daje im potpunu kontrolu nad svojim sredstvima. Stratis puni čvor se može instalirati i pokrenuti na osobnom računalu ili poslužitelju, što programerima i korisnicima omogućuje pristup svim značajkama Stratis lancu blokova, uključujući slanje i primanje kriptovalute STRAX, izvršavanje pametnih ugovora i praćenje stanja mreže. To je važan alat za one koji žele dublje razumjeti i sudjelovati u Stratis ekosustavu.

STRATX - kriptovaluta koja se od 2020 koristi unutar Stratis Platforme. Ovaj token omogućuje plaćanja za transakcije i usluge unutar platforme. Također je korišten za potrebe izgradnje i upravljanja pametnim ugovorima te za interakciju s drugim funkcionalnostima platforme. Prema [18]

Slika 10. Stratis logo (izvor [6])

5. Područja primjene pametnih ugovora

Glavno pitanje koje si postavljamo zašto uopće primjenjujemo pametne ugovore. Od jednostavnih do složenih transakcija, pametni ugovori uklanjaju posrednike i stvaraju neovisnost. Bez obzira na industriju ili scenarij, posrednici uvijek žele svoj udio. S automatiziranim pametnim ugovorom, nije potrebno vjerovati niti plaćati posrednike jer nisu potrebni. To pojednostavljuje postupak i može učiniti pametne ugovore ekonomičnima. Razne industrije počinju prepoznavati svestranost tehnologije pametnih ugovora. Kada su u pitanju stvarni primjeri, posebno u vezi s vlasništvom nekretnina i financijskim uslugama, nema ograničenja. Transakcije su vidljive svima u mreži, stvarajući osjećaj transparentnosti. Također nije moguće mijenjati ili brisati transakcije na distribuiranom registru. Upravo zbog toga, sve više i više primjenjujemo pametne ugovore, a u nastavku ćemo proći neke primjere iz stvarnog svijeta. Prema [14]

5.1 Digitalni identitet

Sve osobne informacije možete pohraniti putem pametnog ugovora kako biste stvorili digitalni identitet. Kada su pametni ugovori povezani s različitim internetskim uslugama, ugovorne strane mogu saznati o pojedincima bez otkrivanja njihovih identiteta. Pametni ugovori mogu također sadržavati kreditne ocjene koje zajmodavci mogu koristiti za procjenu potencijalnih rizika.

Na primjer, MyEarth ID je decentralizirani sustav upravljanja identitetom koji korisnicima omogućuje kontrolu nad svojim podacima o digitalnom identitetu i sigurno njihovo verificiranje s trećim stranama.[14]

5.2 Financije

Decentralizirane financijske (eng. *DeFi*) aplikacije (eng. *dApps*) predstavljaju značajnu alternativu tradicionalnim financijskim uslugama i postaju sve popularnije zahvaljujući svojstvima lanac blokova tehnologije i tehnologije pametnih ugovora. Uz primjenu pametnih ugovora, izbjegavamo treću stranu, odnosno posrednike, koji uz sebe u većini slučajeva vuku određenu dozu nepovjerenja, neizmjenjivost i netransparentnosti. DeFi dApps pružaju usluge slične onima u bankarskoj i financijskoj industriji - poput zajmova, posuđivanja, trgovine i raznih drugih financijskih usluga - zajedno s potpuno novim vrstama proizvoda i decentraliziranim poslovnim modelima koji mogu ponuditi značajne koristi i

praktičnost korisnicima. Zahvaljujući većoj transparentnosti koju omogućuju pametni ugovori (uz 24/7 funkcionalnost i smanjene troškove), dApps imaju potencijal smanjiti prepreke ulasku u svijet financijskih usluga za ljude diljem svijeta i upravo zbog toga, učinci pametnih ugovora u financijskoj industriji sve se više i više osjećaju.

5.3 Zdravstvo

Pametni ugovori imaju potencijal revolucionirati zdravstvenu industriju donoseći transparentnost i učinkovitost zapisivanju podataka. Osim toga, pametni ugovori mogu biti od velike koristi bolničkim bazama podataka, laboratorijima i istraživanjima vezanim uz zdravlje. Oni mogu podržati klinička ispitivanja osiguravajući autentičnost podataka i omogućavajući bolnicama da vode transparentne zapise o podacima pacijenata i učinkovito upravljaju terminima. Kao jedan takav primjer, uzet ćemo Encrypgen, odnosno novonastali projekt koji koristi lanac blokova i pametne ugovore za prijenos podataka DNK. Projekt Gene chain odvaja osobne podatke korisnika od njihovih DNK podataka kako bi ograničio prodaju DNK podataka velikim poduzećima i korisnicima pružio veću kontrolu nad svojim podacima. Prema [19]

5.4 Računalne igre

Lanac blokova i pametni ugovori u svijetu igara su transformirali industriju. Uveli su novi način igranja, poznatiji kao P2E (eng. *play to earn*) igre koje igračima omogućavaju da zarađuju novac dok igraju igre. Osim toga, ove igre su uvele kriptovalute i nezamjenjive tokene (NFT-ove) u igre, što također omogućuje i financijske koristi igračima.

Zadnjih 2,3 godine, P2E igre postale su prilično popularne s porastom brojnih naslova poput The Sandbox, Axie Infinity, Decentraland i drugih. Možete igrati ove P2E igre kako biste zaradili nagrade u njihovim vlastitim tokenima poput MANA-e, SAND-a ili AXS-a. Kripto kovanice koje ove igre nude mogu se koristiti za kupovinu predmeta unutar igre i pretvarati u stvarne predmete

Osim P2E igara, pametni ugovori se također koriste u fantazijskim sportovima koji sve više dobivaju na popularnosti. Kao primjer uzet ćemo igru TradeStars. TradeStars je revolucionarna P2E igra fantazijskih sportova koja koristi tehnologiju lanac blokova kako bi korisnicima pružila nevjerojatno iskustvo igranja. Igrači mogu trgovati dionicama Fantazijskih sportskih igara na TradeStars platformi čija vrijednost direktno utječe na prošle i trenutne stvarne performanse sportaša. TradeStars je razvijen na Ethereum lanac blokova i pokreće ga skalabilno rješenje Layer 2 poznato kao Polygon. Svaka pojedinačna transakcija na

platformi trajno je zabilježena na Ethereumu, potpuno eliminirajući mogućnost bilo kakvog oblika manipulacije ili prijevare. Prema [14]

5.5 Industrija nekretnina

Pametni ugovori također stvaraju veliku razliku u poslovanju sa nekretninama. Bez obzira želimo li kupiti nekretninu ili je iznajmiti, ugovori temeljeni na pametnim ugovorima mogu nam ponuditi siguran i pouzdan način za to. Ova tehnologija bez potrebe za povjerenjem omogućava vam pisanje koda za automatski prijenos vlasništva nad nekretninom ili dogovora o najmu, čime se trgovina nekretninama čini puno sigurnijom osiguravajući da obje strane slijede dogovorene uvjete. Eliminirajući posrednike poput odvjetnika ili posrednika, pametni ugovori mogu smanjiti ukupne troškove registracije i pravne konzultacije te pružiti transparentnost tijekom cijelog procesa.

5.6 Pravna industrija

Pravna industrija je još jedno područje gdje pametni ugovori mogu procvjetati. Mogu se koristiti za obvezujuće pravne ugovore kako za poslovne tako i za društvene aranžmane. Osim toga, pametni ugovori mogu djelovati kao elektronički potpisi za sporazume između različitih strana te smanjiti potrebu za odvjetnicima. Papirnati ili digitalni dokumenti mogu biti krivotvoreni ili izgubljeni u bilo kojem trenutku, što ih čini nedovoljno sigurnima. Sigurnosne revizije pametnih ugovora mogu riješiti taj problem verificiranjem dokumenata na neizmjenjivom distribuiranom registru. Jedna platforma koja primjenjuje tu tehnologiju u pravnim dokumentima je Certoshi. To je sustav izdavanja i verifikacije certifikata koji radi na Ethereum pametnim ugovorima kako bi kriptografski osigurao dokumente.

6. Implementacija aplikacije

U praktičnom primjeru razvit ćemo sustav za glasanje. Sustav za glasanje je kompleksna tema, a obuhvatiti sve aspekte prilično je teško. Naš cilj je prikazati značajke i implementaciju pametnog ugovora. Stoga ćemo izgraditi naš ugovor uzimajući u obzir ograničeni opseg.

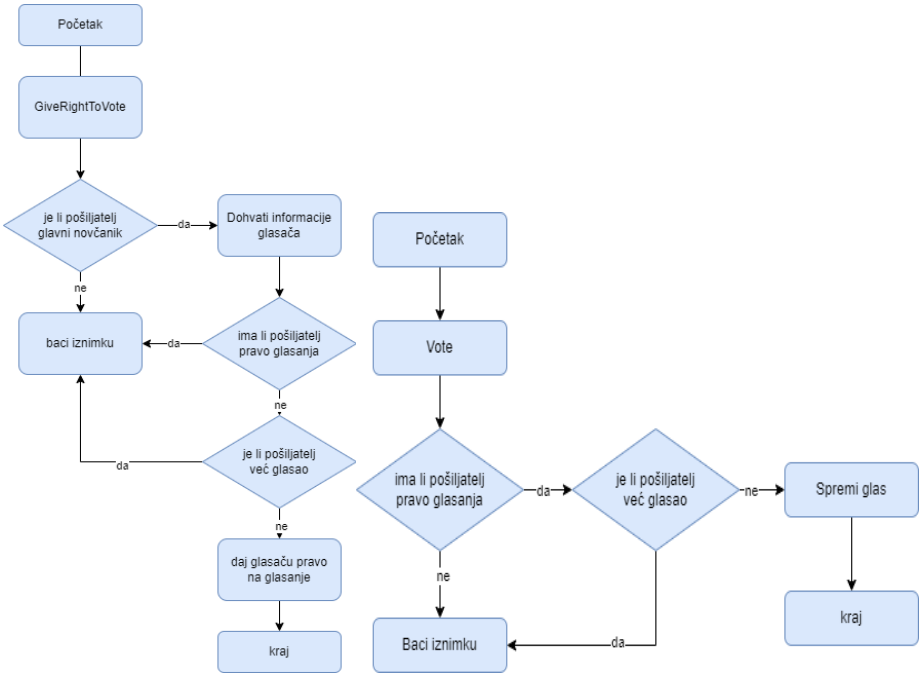
6.1 Implementacija sustava za glasanje

Izradit ćemo pametni ugovor nazvan "Glasovanje". Prilikom implementacije trebao bi prihvatiti popis prijedloga - Prijedlozi mogu biti bilo što, poput kandidata ili opcija ankete. Novčanik koji potječe od ugovora postat će glavni novčanik. Glavni novčanik može dodijeliti pravo drugim novčanicima (korisnicima) da glasaju. Birač s pravom glasovanja može pozvati metodu "Glasaj" i registrirati svoj glas. Prijedlog koji dobije najviše glasova treba se smatrati pobjedničkim prijedlogom. Svako može pozvati metodu kako bi saznao pobjednika glasovanja.

6.1.1 Scenariji korištenja

Kako bi bolje razumjeli pametni ugovor, proći ćemo kroz sljedeće korake:

- Postavljanje ugovora
- Struktura ugovora
- Kreiranje testnog projekta
- Validacija i kompilacija pametnog ugovora
- Postavljanje Stratis blockchaina
- Učitavanja Swagger-a
- Postavljanje Postman-a
- Učitavanje novčanika
- Dohvaćanje adresa novčanika za testiranje ugovora
- Dohvaćanje potvrde transakcije
- Dodjela prava glasanja
- Glasanje
- Dohvaćanje razultata glasanja



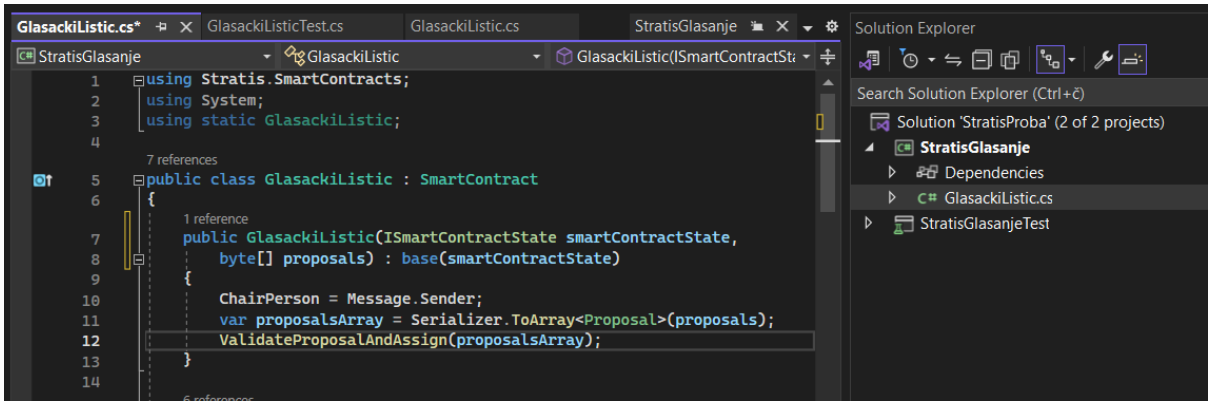
Slika 11: Dijagram toka za metodu GiveRightToVote i metodu Vote (izvor: vlastita izrada)

6.1.1.1. Postavljanje ugovora

Prvo u Visual studiu uz pomoć Stratis template-a kreiramo novi projekt. Predložak za Stratis pametni ugovor je projekt predložak za Visual Studio koji stvara uzorak Stratis pametnog ugovora u okruženju Visual Studio. Da biste ga instalirali, koristite sljedeću naredbu u naredbenom prozoru programa Visual Studio:

```
dotnet new --install Stratis.SmartContracts.Templates.CLI
```

Nakon uspješne izvedbe ove naredbe, predložak projekta za Stratis pametne ugovore bit će instaliran za Visual Studio. Kada se instalacija uspješno završi, ovaj predložak će biti dostupan u popisu dostupnih predložaka. Nazvat ćemo ga „StratisGlasanje“. Prva linija u ugovoru je referenca na Stratis.SmartContracts NuGet paket. Ovaj paket omogućava nasljeđivanje iz klase "SmartContract". Na taj način možemo koristiti korisnu funkcionalnost kao što su slanje sredstava i spremanje podataka. Nadalje, koristimo framework okvir na 3.1. To će nam pomoći da izbjegnemo probleme s verzijama paketa tijekom razvoja kasnije. Preimenovali smo naziv pametnog ugovora i dali mu simboličan naziv „GlasackiListic“. Također za kreiranje Stratis pametnih ugovora, moramo provjeriti da je pametni ugovor ispravan. Proces provjere pametnog ugovora obavlja se pomoću alata sct koji pruža Stratis te smo ga preuzeli preko github poveznice [13]. Proces provjere je obavezan kako bismo provjerili valjane uvjete korištene u pametnom ugovoru. Proces osigurava da je pametni ugovor oslobođen nedeterminističkih elemenata i da su ispunjeni dodatni uvjeti vezani uz format ugovora.



Slika 12: postavljanje pametnog ugovora (desktop screenshot)

6.1.1.2. Struktura ugovora

Nadalje treba nam Struktura koja će predstavljati pojedinog birača. Ona pohranjuje informacije o biraču poput težine glasovanja, je li birač glasao ili nije i indeks prijedloga na koji je birač glasao. Također potrebna nam je struktura prijedloga koja će sadržavati informacije o prijedlozima i broju glasova za svaki prijedlog.

```
public struct Voter
{
     public uint Weight;
     public bool Voted;
     public uint VoteProposalIndex;
 }
 public struct Proposal
{
     public string Name;
     public uint VoteCount;

 }
```

Također nam je potrebno imati Get i Set metode za obje strukture koje smo definirali. Za pohranu podataka u pametni ugovor koristit ćemo PersistentState. PersistentState pohranjuje podatke na ključ, i isti ključ moramo koristiti za dohvaćanje podataka. Prijedlozi bi trebali biti pohranjeni samo jednom prilikom implementacije ugovora. I nismo sigurni koliko će prijedloga bilo tko dodati; stoga koristimo dinamički niz za pohranu prijedloga.

```
     private Voter GetVoter(Address address) =>
PersistentState.GetStruct<Voter>($"voter:{address}");
private void SetVoter(Address address, Voter voter) =>
PersistentState.SetStruct($"voter:{address}", voter);
```

```
public Proposal[] Proposals
{
    get => PersistentState.GetArray<Proposal>(nameof(Proposals));
    private set => PersistentState.SetArray(nameof(Proposals), value);
}
public Address MainWallet
{
    get => PersistentState.GetAddress(nameof(MainWallet));
    private set => PersistentState.SetAddress(nameof(MainWallet),
value);

}
```

Prilikom implementacije ugovora proslijedit ćemo popis prijedloga što također treba pohraniti. Uzimamo bajtni niz kao parametar u konstruktoru, a zatim ga serijaliziramo u niz prijedloga.

```
public GlasackiListic(ISmartContractState smartContractState,byte[]
proposals) : base(smartContractState){
    MainWallet = Message.Sender;
    var proposalsArray = Serializer.ToArray<Proposal>(proposals);
    ValidateProposalAndAssign(proposalsArray);
}
```

Kada uzimamo unos od korisnika, bolje je provjeriti valjanost prije nego što obavimo bilo koju operaciju. To možemo postići pomoću pomoćne metode. Metodu smo nazvali „ValidateProposalAndAssig „ s kojom provjeravamo duljinu niza prijedloga. Ako je duljina niza ispravna, dodijelimo niz prijedloga svojstvu 'Proposal'.

```
private void ValidateProposalAndAssign(Proposal[] proposals)
{
    Assert(proposals.Length > 1, "Molimo dostavite minimalno 2
prijedloga");
    this.Proposals = proposals;
}
```

Sada dodajmo metodu za dodjelu prava glasovanja korisniku. Kao što smo ranije spomenuli, samo glavni novčanik može pozvati ovu metodu.

```
public bool GiveRightToVote(Address voterAddress)
{
    Assert(Message.Sender == MainWallet, "samo glavni novčanik ima pravo
na dodjeljivanje prava za glasanje.");
    var voter = this.GetVoter(voterAddress);
    Assert(voter.Weight == 0, "glasač već ima prava.");
    Assert(!voter.Voted, "već glasao.");
```

```
        voter.Weight = 1;
        this.SetVoter(voterAddress, voter);
        return true;
    }
```

Dodajemo još jednu metodu za glasovanje. Glasač će proslijediti indeks prijedloga, a metoda će zabilježiti glas. Također, metoda ažurira niz prijedloga i povećava broj glasova.

```
        public bool Vote(uint proposalId)
     {
         var voter = this.GetVoter(Message.Sender);
         Assert(voter.Weight == 1, "Has no right to vote.");
         Assert(!voter.Voted, "Already voted.");
         voter.Voted = true;
         voter.VoteProposalIndex = proposalId;
         Proposals[proposalId].VoteCount += voter.Weight;
         this.SetVoter(Message.Sender, voter);
         Log(new Voter
         {
             Voted = true,
             Weight = 1,
             VoteProposalIndex = proposalId
         });
         return true;
     }
```

Assert je metoda klase „SmartContracts“, koristi se za validaciju, a sintaksa joj je sljedeća:

```
Assert(bool condition, string message = "neuspjesno.");
```

Dodajemo još jednu metodu koja broji broj glasova i vraća prijedlog sa najviše dobivenih glasova.

```
public uint WinProposal()
{
    uint winVoteCount = 0;
    uint WinProposalId = 0;

    for (uint i = 0; i < Proposals.Length; i++)
    {
        if (Proposals[i].VoteCount > winVoteCount)
        {
            winVoteCount = Proposals[i].VoteCount;
```

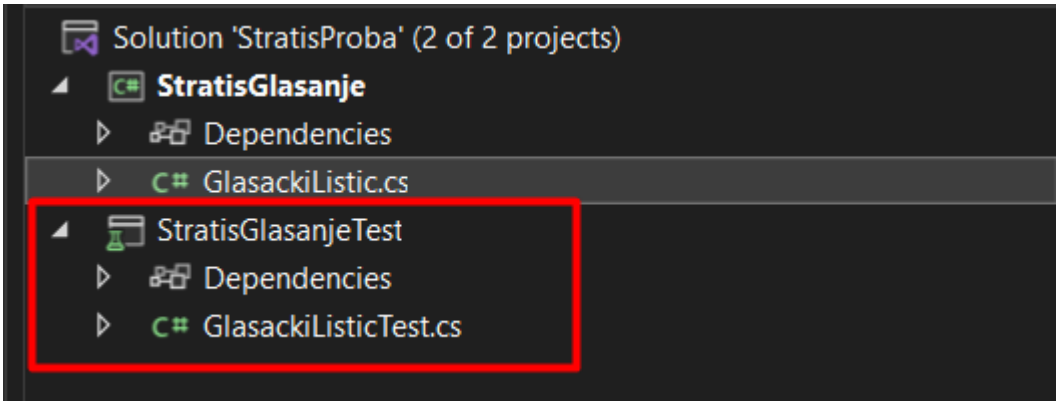
```
                WinProposalId = i;
            }
        }
        return WinProposalId;
    }
```

Slična prošloj metodi, da bismo dobili naziv pobjedničkog prijedloga, dodajemo još jednu metodu, koja će koristiti metodu "WinProposal" i dohvatiti naziv prijedloga.

```
public string WinnerName()
{
    var WinProposalId = WinProposal();
    var proposals = Proposals[WinProposalId];
    return proposals.Name;
}
```

6.1.1.3. Kreiranje testnog projekta

Nadalje dodat ćemo novi xUnit Test projekt i nazvat ćemo ga „StratisGlasanjeTest“. Koristit ćemo .NET core 3.1 Framework, isti kao i kod glavnog projekta. Zatim preko konzole instaliramo potrebne nuGet pakete i dodajemo referencu na projekt pametnog ugovora.



Slika 13: testni projekt (desktop screenshot)

Stvaramo niz prijedloga s dvije evidencije, serijaliziramo ga i stvaramo heksadekadski niz.

```
namespace StratisGlasanjeTest.Tests
{
    using Moq;
    using NBitcoin;
    using Stratis.SmartContracts.CLR.Serialization;
    using Stratis.SmartContracts.Core;
    using Xunit;
    using Xunit.Abstractions;
```

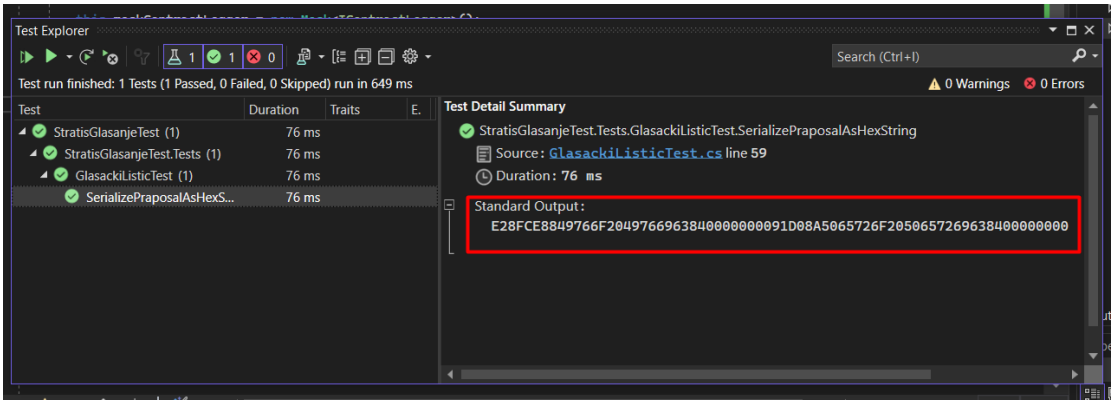
```
    using Proposal = Ballot.Proposal;

    public class GlasackiListicTest
     {
        private readonly ITestOutputHelper testOutputHelper;
        private Serializer;
        private Mock < Network > network;

        public GlasackiListicTest (ITestOutputHelper testOutputHelper)
            {
            this.testOutputHelper = testOutputHelper;
            this.network = new Mock < Network > ();
            this.serializer = new Serializer(new
ContractPrimitiveSerializerV1(this.network.Object));
        }

        [Fact]
        public void HexStringSerializer()
            {
            var proposals = new [] {
                new Proposal { Name = "Ivo Ivic", VoteCount = 0 },
                new Proposal { Name = "Pero Peric", VoteCount = 0 }
            };

this.testOutputHelper.WriteLine(this.serializer.Serialize(proposals).ToH
exString());
        }
    }
}
```



Slika 14. provedba testa (desktop screenshot)

Nakon uspješne provedbe testa, u crveno označenom pravokutniku, vidljivi su parametri koje trebamo proslijediti prilikom implementacije pametnog ugovora.

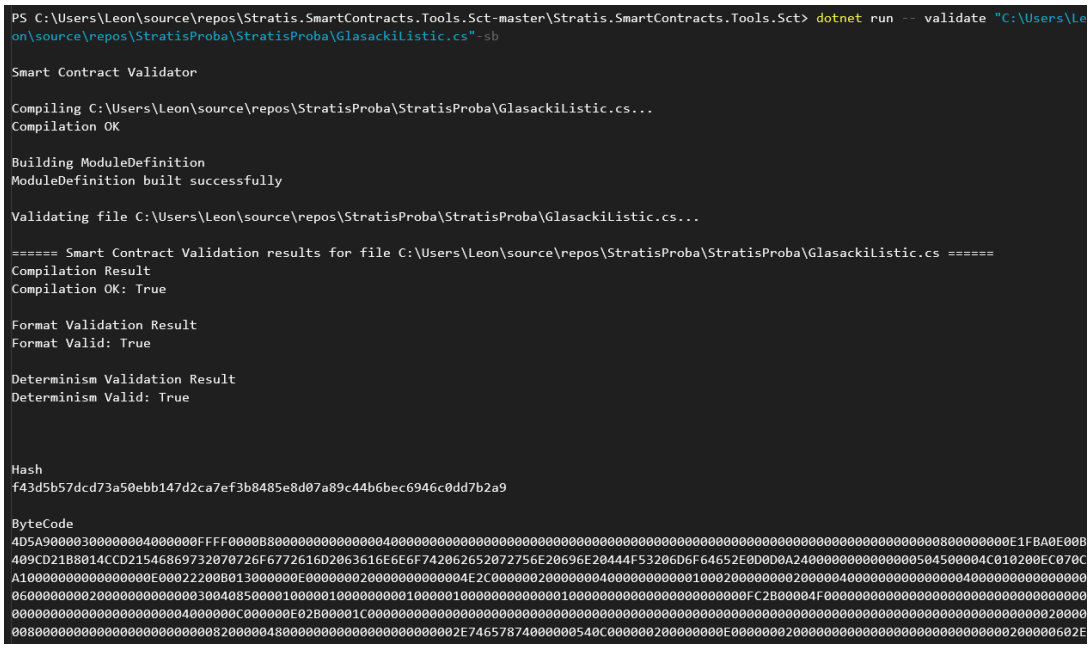
6.1.1.4. Validacija i kompilacija pametnog ugovora

Nadalje ćemo provjeriti, odnosno validirati i kompilirati naš ugovor pomoću alata „Sct tool“. Ukoliko je ugovor valjan, dobit ćemo bajtkod ugovora koji ćemo primijeniti prilikom izvođenja ugovora. Validaciju provodimo u terminalu projekta „Stratis.SmartContract.Tools.Sct“ kojeg smo preuzeli ranije prilikom pripreme projekta.

```
dotnet run -- validate [CONTRACT_PATH_HERE]
```

```
dotnet run -- validate [CONTRACT_PATH_HERE] -sb
```

Prva naredba koju ćemo izvršiti je naredba za validaciju pametnog ugovora. Nakon što smo izvršili validaciju, odnosno provjeru ugovora, moramo kompilirati kod pametnog ugovora i generirati bajtkod. Taj bajtkod pametnog ugovora je kod koji moramo implementirati na lanac blokova. Nakon uspješne kompilacije, bit će nam prikazan heksadecimalni zapis i bajtkod u terminalu, što je i prikazano na donjoj slici.



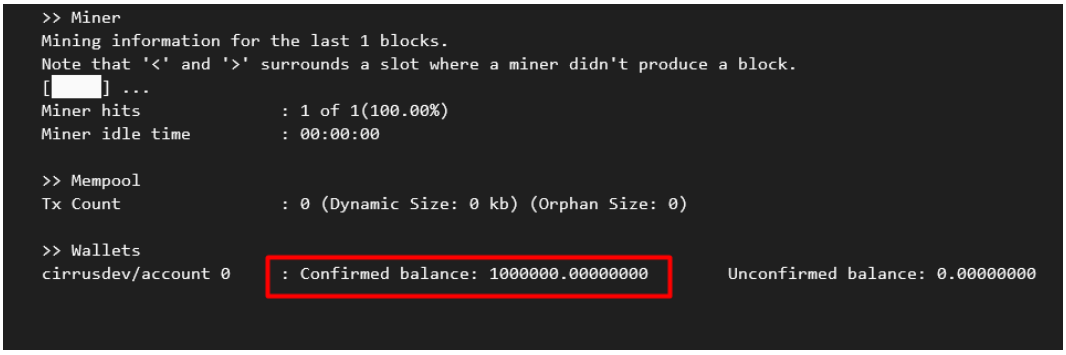
Slika 15: provjera pametnog ugovora (desktop screenshot)

6.1.1.5. Postavljanje Stratis blockchaina

Sada dolazi najvažniji dio. Trebamo instalirati Stratis lanac blokova na lokalnom računalu. Lokalni lanac blokova je kopija stvarnog lanca blokova, ali koristi drugi mehanizam konsenzusa. Kako bi olakšala razvoj i učenje, omogućuje jednostavno izvođenje i testiranje pametnih ugovora na razvojnom računalu bez potrebe za vanjskim rudarima. Da bismo pokrenuli lanac blokova na lokalnom računalu u svrhu razvoja, moramo preuzeti Stratis full

node i u preuzetom projektu pokrenuti terminal u mapi „Stratis.CirrusMinerD“. U terminalu napišemo navedenu komandu.

```
dotnet run -devmode=miner
```

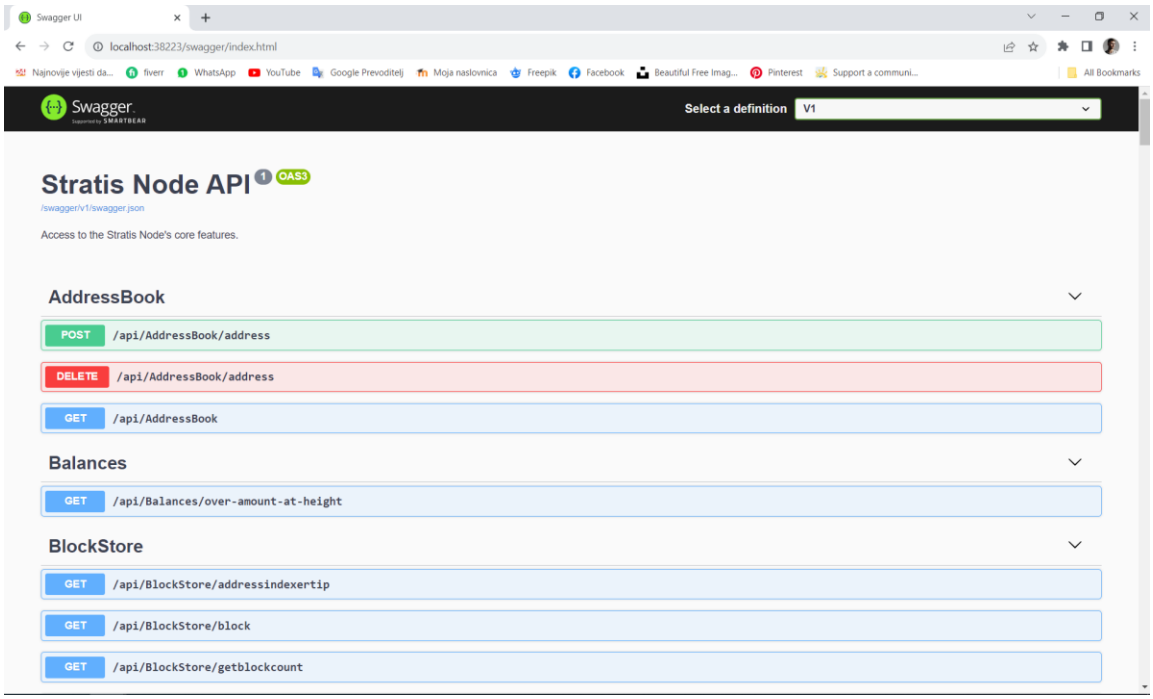


Slika 16: pokretanje Stratisa (desktop screenshot)

Gornja naredba pokreće pojedinačni čvor za proizvodnju blokova koristeći PoA (Proof-of-Authority) algoritam konsenzusa.

6.1.1.6. Učitavanja Swagger-a

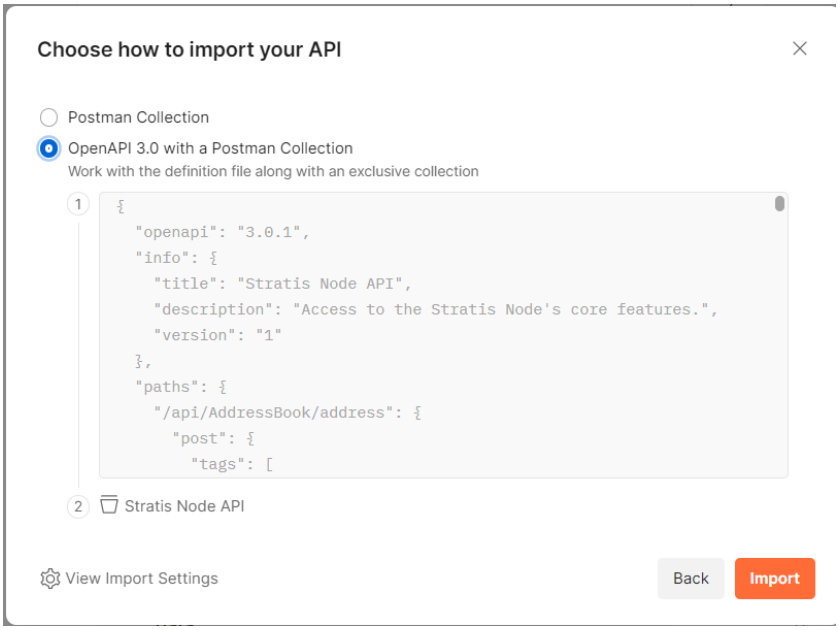
Nakon uspješne izvedbe gornje naredbe otvaramo *http://localhost:38223/swagger* na svom računalu gdje možete vidjeti popis API-ja. Ovi API krajnjih točaka koristit će se za interakciju s lancem blokova.



Slika 17: Swagger početni zaslon (desktop screenshot)

6.1.1.7. Učitavanja Postmana

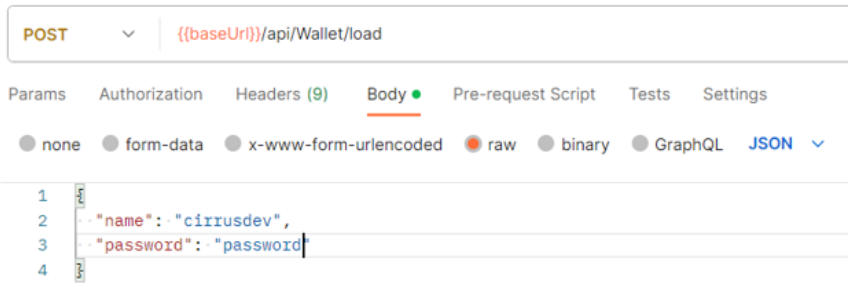
Postman je popularan klijent za API-je koji se koristi za pozivanje API krajnjih točaka. Da bismo koristili Postman, trebamo se registrirati i kreirati račun. Koristit ćemo Postman za implementaciju našeg ugovora i izvođenje metoda ugovora. Kako bismo dobili sve dostupne metode sa Swagger stranice u Postman, moramo preuzeti .json datoteku u Swaggeru i uvesti u Postman.



Slika 18: uvoz .json datoteke u Postman (desktop screenshot)

6.1.1.8. Učitavanje novčanika

Prvo moramo prenijeti vrijednosti unaprijed definiranog novčanika u API za učitavanje novčanika

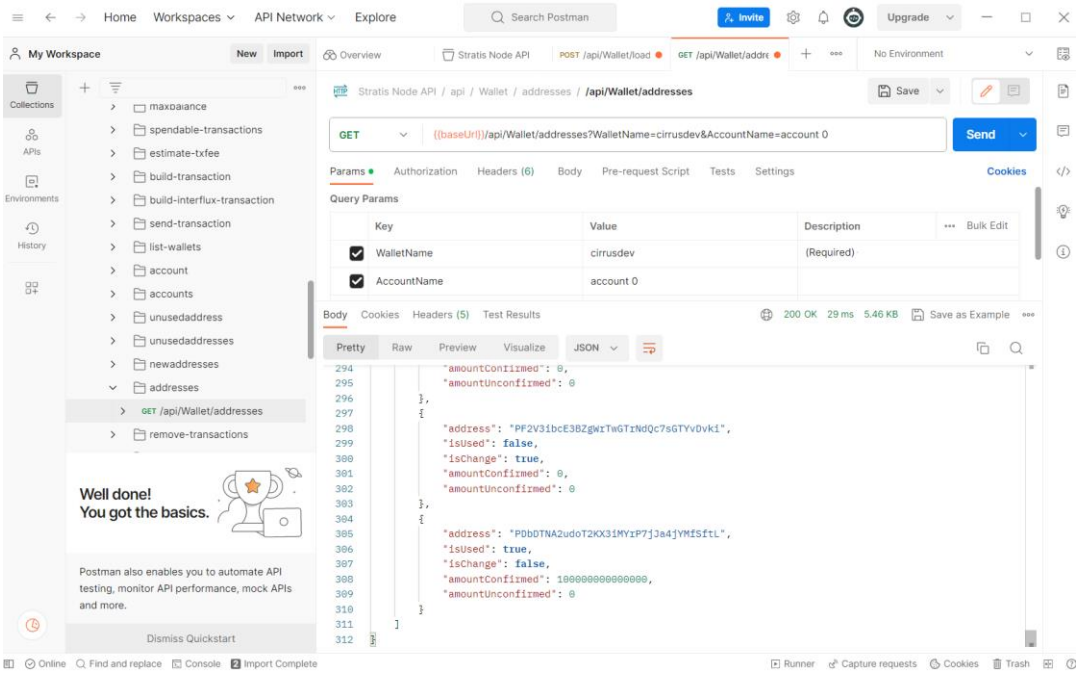


Slika 19: učitavanje novčanika (desktop screenshot)

6.1.1.9. Dohvaćanje adresa novčanika za testiranje ugovora

Za ugovor su potrebne određene adrese novčanika za glavnog novčanika i ostale novčanike, odnosno glasače. Adrese koje koristimo trebaju imati određeni saldo kako bi se

mogla izvršiti transakcija. Da bismo dobili te adrese, upotrijebiti ćemo API endpoint: /api/Wallet/balance. Ovaj endpoint vraća adrese novčanika zajedno s njihovim saldom

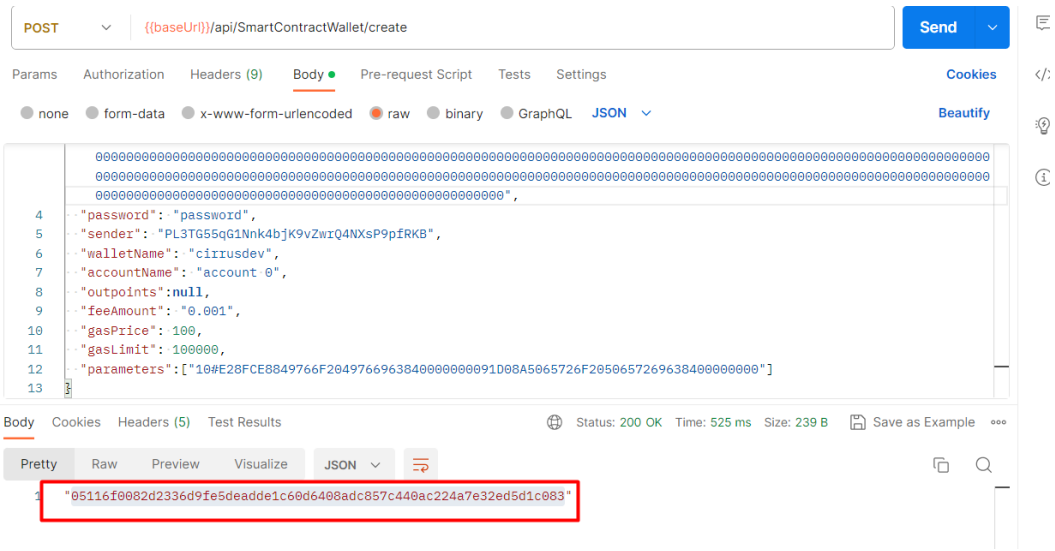


Slika 20: provjera salda (desktop screenshot)

Kao što vidimo, postoje adrese koje imaju pozitivan saldo i upravo te adrese ćemo koristiti. Ako kojim slučajem samo jedna adresa sadrži pozitivan saldo, možemo prebaciti s zadane adrese koristeći „splitcoin“. To smo i učinili te tako dobili 4 adrese sa pozitivnim saldom.

- PL3TG55qG1Nnk4bjK9vZwrQ4NXsP9pfRKB – glavna adresa
- PMmsQsTEdFcdukLicVonjw3VxQHcksWq3H – glasač 1
- PGb3v8ZhCiXHFsGSPV9r96ezSd7X2UMX9b – glasač 2
- PDbDTNA2udoT2KX3iMYrP7jJa4jYMfSftL – glasač 3

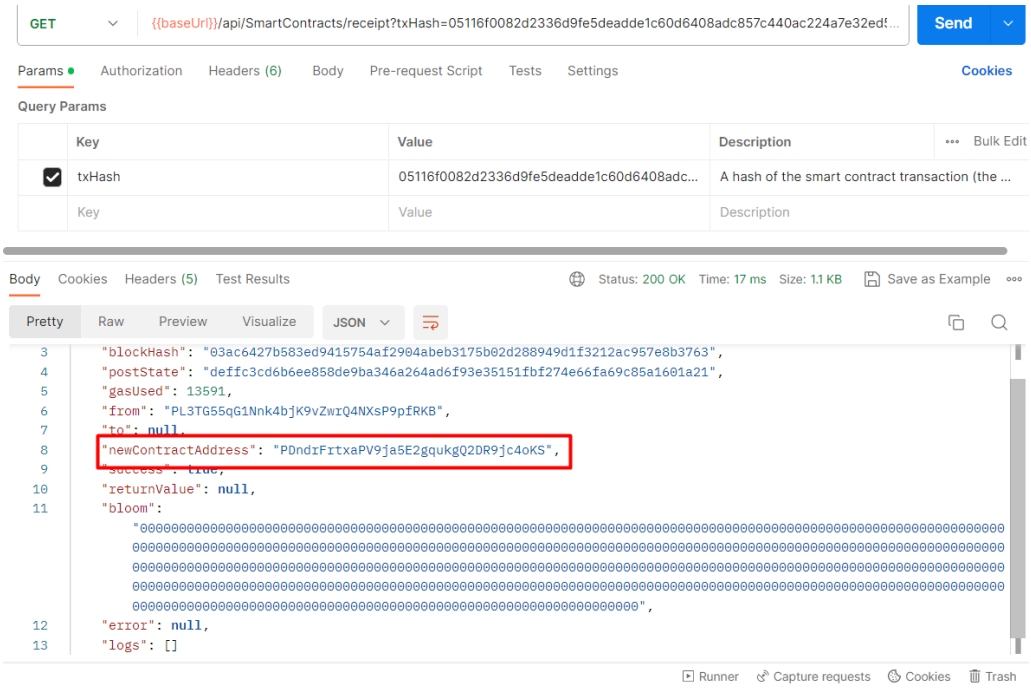
Prilikom implementacije ugovora, moramo prenijeti bajtkodove generirane prethodno pomoću alata „Sct tool“ te također proslijediti parametre koje smo dobili prilikom provođenja testova.



Slika 21: provedba pametnog ugovora (desktop screenshot)

6.1.1.10. Dohvaćanje potvrde transakcije

Pomoću heksadecimalnog zapisa transakcije možemo dohvatiti potvrdu transakcije koja sadrži informacije o potrošenom plinu, adresama pošiljatelja i primatelja, povratnim vrijednostima, dnevnicima itd. Kao parametar prilikom upita unosimo heksadecimalni zapis transakcije. Ovo ćemo često koristiti kako bismo dobili rezultate svih transakcija koje obavljamo na lancu blokova, kao što su dodjela prava glasovanja ili glasovanje o prijedlozima



Slika 22: dobivanje adrese ugovora (desktop screenshot)

Gore označenu adresu koristit ćemo prilikom interakcije s implementiranim pametnim ugovorom.

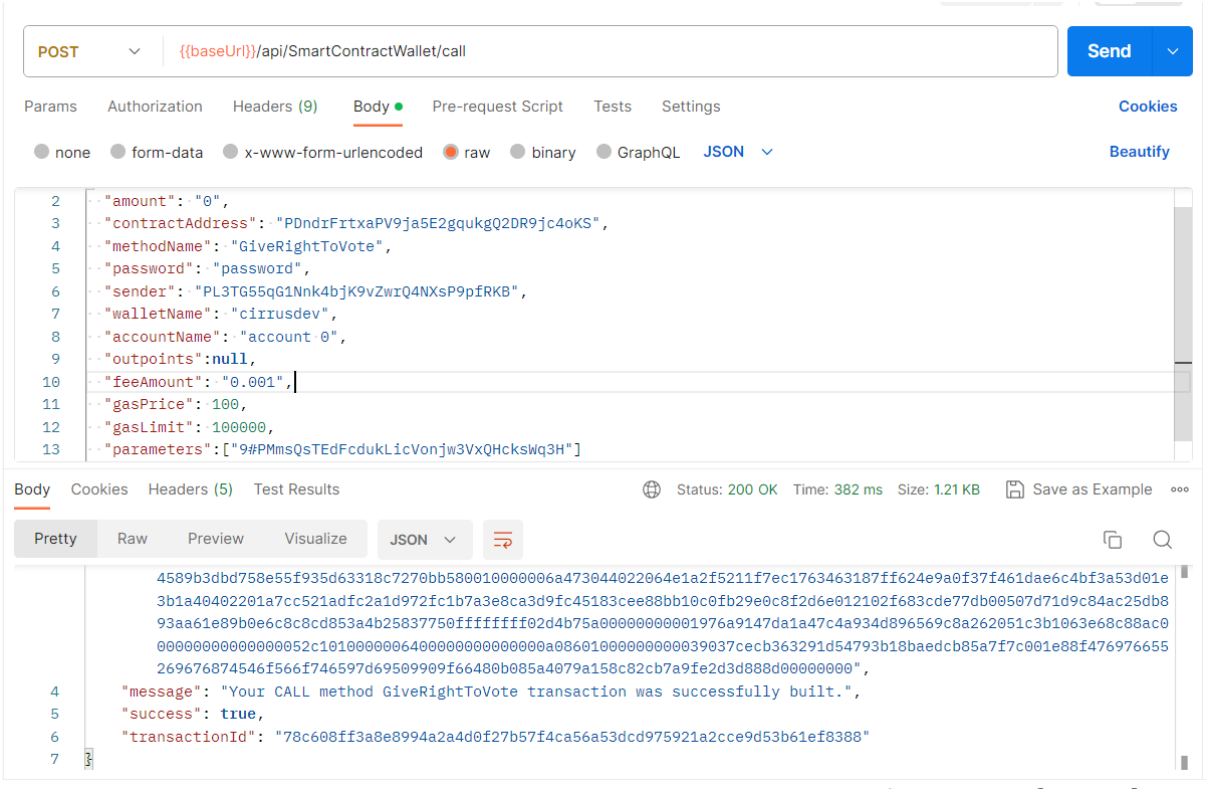
6.1.1.11. Dodjela prava glasovanja

Postoje dvije metode za interakciju s pametnim ugovorom. Ako želimo promijeniti stanje pametnog ugovora, moramo koristiti poziv ugovora, dok za čitanje podataka koristimo lokalni poziv. Stoga ćemo ovdje koristiti poziv ugovora kako bismo omogućili pravo glasovanja adresi novčanika. Napomenimo da samo glavni novčanik može dodijeliti pravo glasovanja drugoj adresi novčanika. Koristit ćemo metodu pametnog ugovora putem odgovarajuće pozivne točke. Unosimo sljedeće vrijednosti:

Tablica 1: Prikaz podataka za poziv metode „GiveRightToVote“

Parametar	Vrijednost	Opis
Iznos	0	Iznos STRAX-a koje šaljemo ugovoru
Adresa ugovora	PDndrFrtxaPV9ja5E2gqukgQ2DR9jc4oKS	Adresa ugovora
Naziv metode	GiveRightToVote	Naziv metode
lozinka	password	Lozinka novčanika
pošiljatelj	PL3TG55qG1Nnk4bjK9vZwrQ4NXsP9pfRKB	Adresa pošiljatelja
Naziv novčanika	cirrusdev	Naziv novčanika
Naziv korisnika	account 0	Naziv korisnika
outpoints		null
feeAmount	0.001	Iznos naknade
gasPrice	100	Cijena „goriva“
gasLimit	100000	Limit „goriva“
parametri	["9# PMmsQsTEdFcdukLicVonjw3VxQHcksWq3H"]	Parametri koje šaljemo metodi ugovora. “9#{glasač1_adresa}”

(Izvor: vlastita izrada)



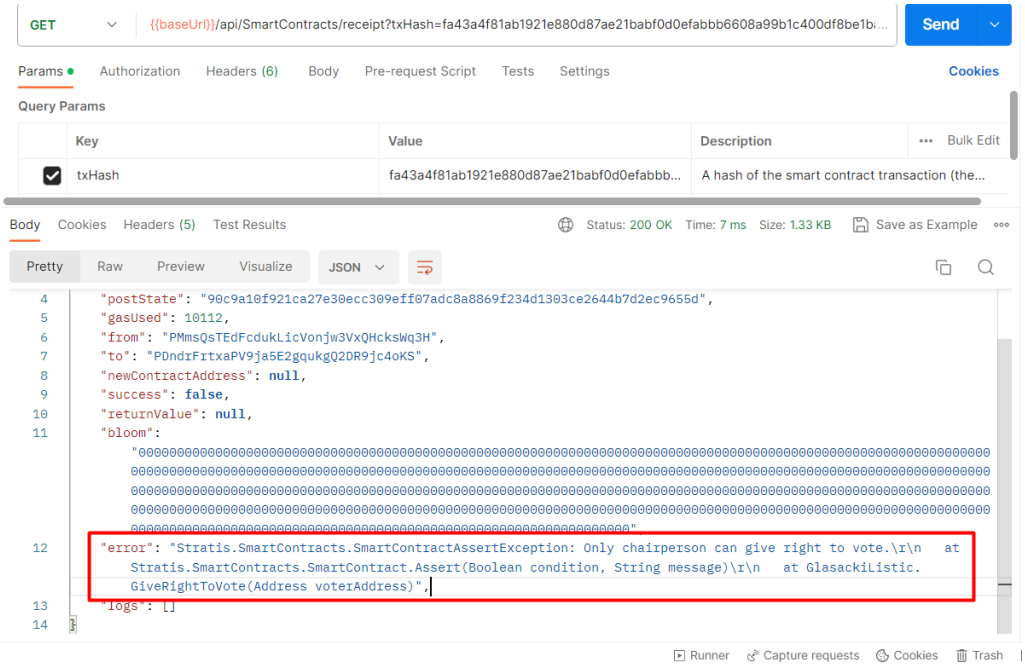
Slika 23: poziv pametnog ugovora za davanje prava glasanja (desktop screenshot)

Opet dobivamo identifikacijski broj transakcije kao odgovor. Međutim, kako bi bili sigurni da li je naša metoda uspješno izvršena, trebamo izvršiti poziv za potvrdu transakcije. Kopiramo identifikacijski broj transakcije i prenesimo ga na poziv za dobivanje potvrde. Kao što smo prije napomenuli, imamo tri glasača i do sada pravo glasovanja smo dali samo jednom od njih. Da bismo omogućili pravo glasovanja i druga dva glasača, ponoviti ćemo isti postupak.

Slika 24: provjera ispravnosti metode (desktop screenshot)

Odgovor nam ne daje nikakve greške što znači da je validacija uspješno prošla.

Sada ćemo izvršiti poziv prošle metode, ali ovaj put postavljamo glasača 2 (tj. PGb3v8ZhCiXHFsGSPV9r96ezSd7X2UMX9b) kao pošiljatelja. Dobit ćemo identifikacijski broj transakcije. Taj ID ćemo spremiti i dohvatite potvrdu transakcije. Trebali bismo primijetiti da je vrijednost parametra "uspjeh" postavljena na "false" i da će se prikazati poruka koju smo postavili prilikom provjere.



Slika 25: provjera prava glasanja (desktop screenshot)

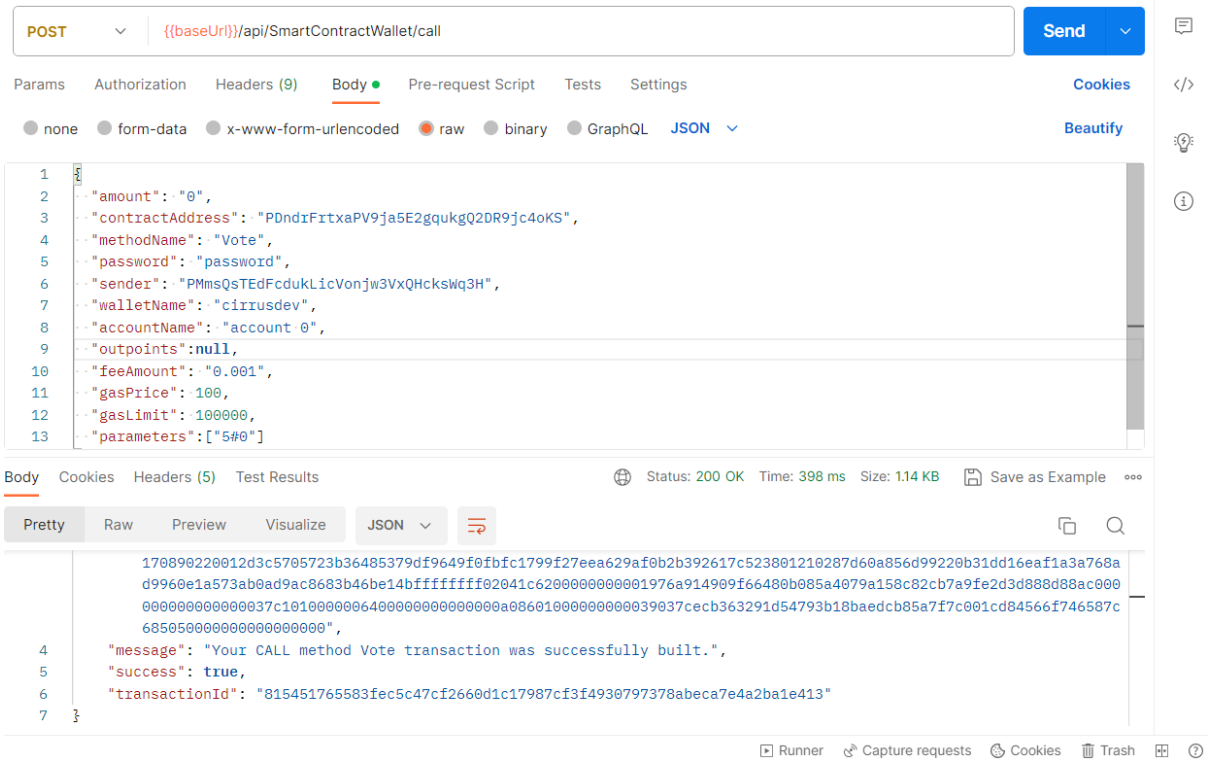
6.1.1.12. Glasanje

Glasač s ispravnim pravom glasovanja može izvršiti poziv metode "Vote". Za poziv metode koristimo isti poziv, samo mijenjajući parametre. Isti slučaj ponovimo za sva 3 glasača.

Tablica 2: Prikaz podataka za poziv metode „Vote“

Parametri	Vrijednost	Opis
Naziv metode	Vote	Naziv metode
pošiljatelj	PMmsQsTEdFcdukLicVonjw3VxQHcksWq3H	Adresa glasača 1
parametri	["5#0"]	Indeks glasanja. "5#{indeks_glasanja}". Prijedlozi su u nizu, pa je indeks prvog prijedloga 0, a za drugi je 1."

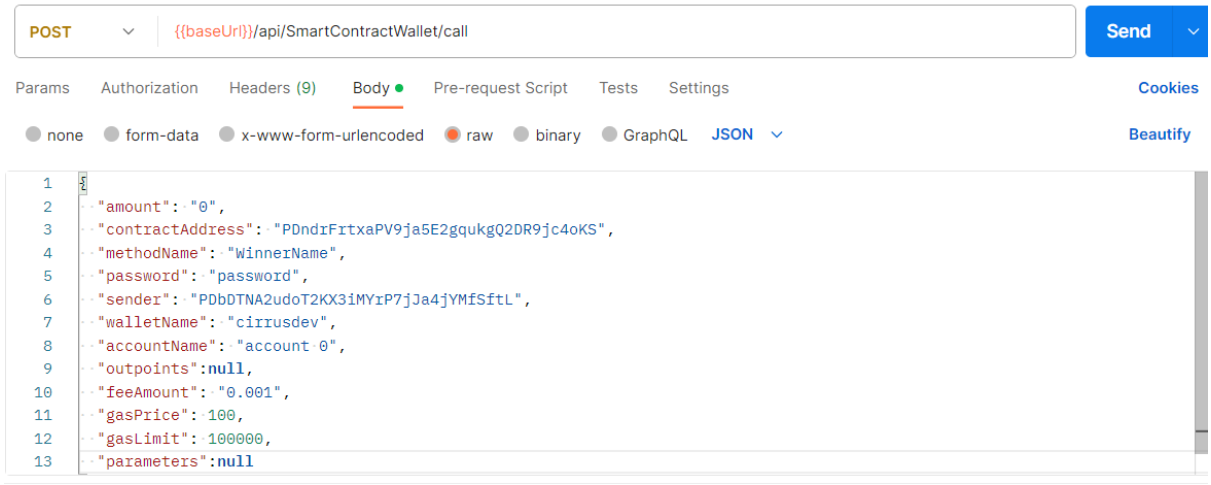
(Izvor: vlastita izrada)



Slika 26: provedba metode „Vote“ (desktop screenshot)

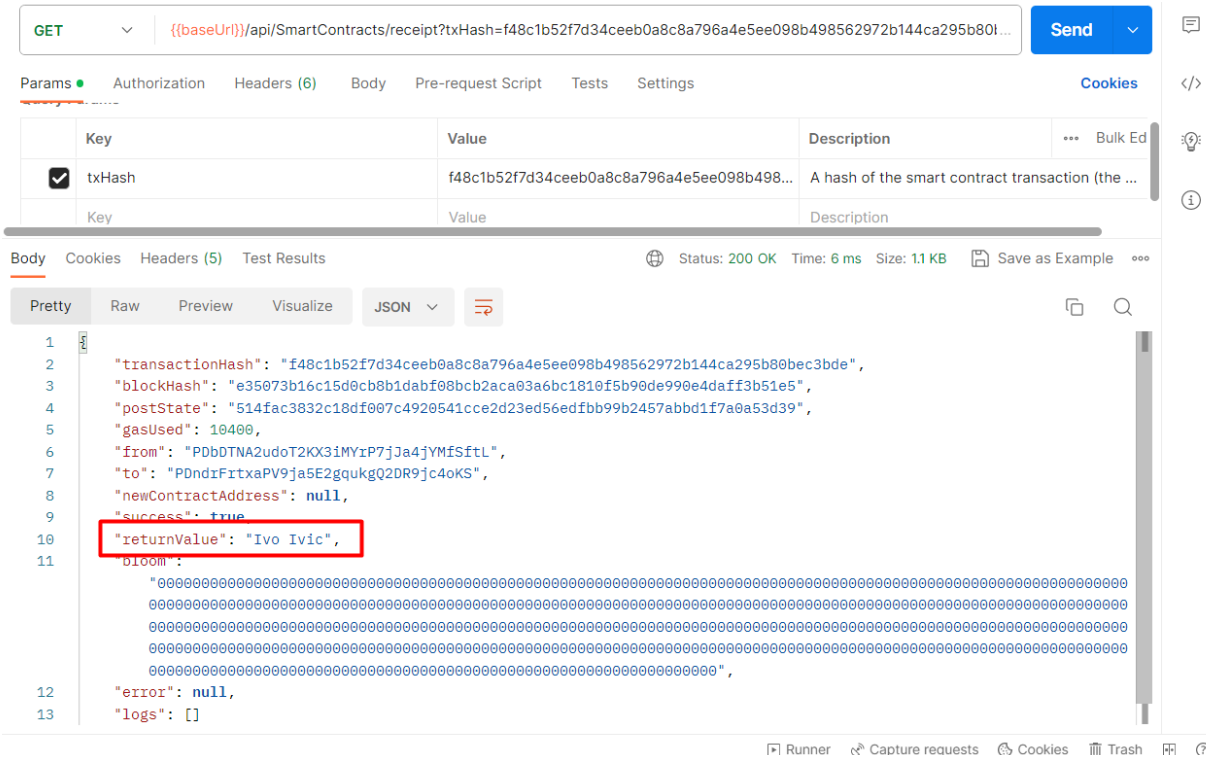
6.1.1.13. Dohvaćanje rezultata glasanja

Sada pozivamo metodu "WinnerName". Svaka osoba može pozvati ovu metodu, stoga možemo koristiti bilo koju adresu novčanika koja ima saldo. Mi ćemo kao primjer koristiti adresu posljednjeg glasača.



Slika 27: provedba metode „WinnerName“ (desktop screenshot)

Poziv nam kao rezultat vraća identifikacijski broj transakcije koji treba unijeti u poziv za potvrdu pametnog ugovora kako bismo dobili rezultat.



Slika 28: provjera pobjednika glasanja (desktop screenshot)

S obzirom da smo dali 2 glasa prijedlogu s indeksom 0 i jedan glas prijedlogu s indeksom 1, prijedlog 1 pod nazivom „Ivo Ivic“ je pobjednik glasanja.

7. Zaključak

Razvoj pametnih ugovora unutar lanca blokova predstavlja značajnu inovaciju u digitalnom poslovnom svijetu. Ovi autonomni ugovori omogućuju automatizaciju transakcija i pružaju visoku razinu transparentnosti i integriteta podataka. Unatoč svojim brojnim prednostima, razvoj pametnih ugovora također nosi izazove, uključujući potrebu za pažljivom sigurnosnom analizom i temeljnim testiranjem koda. Lanac blokova i pametni ugovori već mijenjaju različite sektore, uključujući financijski, logistički i zdravstveni. Kako se tehnologija dalje razvija, očekuje se da će imati sve veći utjecaj na način na koji poslujemo i komuniciramo. Razvoj pametnih ugovora predstavlja uzbudljiv korak prema budućnosti poslovanja koja je decentralizirana i iznimno efikasna. S porastom interesa za pametnim ugovorima, važno je naglasiti da njihova uspješna primjena zahtijeva suradnju između stručnjaka za tehnologiju, pravne struke i inovatora u poslovanju kako bi se osigurala njihova potpuna legalna i funkcionalna valjanost. Očekuje se da će daljnji napredak tehnologije i razvoj regulativnih okvira potaknuti širenje upotrebe pametnih ugovora u globalnom poslovanju. Za implementaciju našeg praktičnog dijela koristili smo Stratis, koji također pruža alate i resurse kako bi podržao programere tijekom razvoja i testiranja njihovih pametnih ugovora. Spoj Stratis platforme, C# programskog jezika i pametnih ugovora otvara vrata inovacijama u blockchain tehnologiji i obećava široku primjenu u stvarnim poslovnim scenarijima.

8. Literatura

[1] Rangel Stoilov (2019). Solidity Smart Contracts: Build DApps In The Ethereum Blockchain,

[2] Imran Bashir (2017). Mastering Blockchain , Packt Publishing

[3] Kurt Dugan (2018). Blockchain for Beginners: Understand the Blockchain Basics and the Foundation of Bitcoin and Cryptocurrencies, CRB Publishing

[4] Nick Woods (2019). Ethereum for Beginners: The Complete Guide to Understanding Ethereum, Blockchain, Smart Contracts, ICOs, and Decentralized Apps, Publishing Forte

[5] Dr. Hedaya Mahmood Alasooly (2020). Quick Guide for Smart Contracts Creation and Deployment on Ethereum Blockchain, BookRix

[6] Sebastian Andres (2021). Ethereum in a Nutshell: The Definitive Guide to Enter the World of Ethereum, Cryptocurrencies, Smart Contracts and Master It Completely: Cryptocurrency Basics, #2“, W Publishing

[7] „What is Blockchain? Everything You Need to Know“ (2021). [Na internetu]. Preuzeto: https://www.simplilearn.com/tutorials/Blockchain a-tutorial/what-is-blockchain #what_is_blockchain [Pristupano: 18.08.2023]

[8] „What is Blockchain Technology and How Does it Work?“ (2020). [Na internetu]. Preuzeto: https://www.creative-tim.com/blog/educational-tech/what-is-blockchain - technology-and-how-does-it-work/ [Pristupano: 19.08.2023]

[9] „Smart Contracts in Blockchain“ (2022). [Na internetu]. Preuzeto: https://www.shiksha.com/online-courses/articles/smart-contracts-in-blockchain /#s1 [Pristupano: 20.08.2023]

[10] „Blockchain structure“ (2022). [Na internetu]. Preuzeto: https://www.geeksforgeeks.org/ blockchain -structure/ [Pristupano. 21.08.2023]

[11] „What Do We Mean by Smart Contracts? Open Challenges in Smart Contracts“ (2020). [Na internetu]. Preuzeto: https://www.frontiersin.org/articles/10.3389/fbloc.2020.553671/full [Pristupano: 13.04.2023]

[12] Nick Szabo, „Smart Contracts“ (1994). [Na internetu] Preuzeto: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html [Pristupano: 13.04.2023]

[13] https://github.com/stratisproject/Stratis.SmartContracts.Tools.Sct

[14] „10 Real-World Smart Contract Use Cases“ (2021). [Na internetu] Preuzeto: https://hedera.com/learning/smart-contracts/smart-contract-use-cases [Pristupano: 10.09.2023]

[15] „Top 7 smart contract platforms to consider in 2023“ (2023). [Na internetu] Preuzeto: https://www.techtarget.com/searchcio/tip/Top-smart-contract-platforms-to-consider Pristupano [10.06.2023]

[16] „Types of Blockchain: Public, Private, or Something in Between“. (2021). [Na internetu] Preuzeto: https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between Pristupano [10.06.2023]

[17] Daniel Drescher, „Blockchain Basics“, (2017). [Na internetu] Preuzeto: https://app.scnu.edu.cn/iscnu/learning/block_chain/Blockchain%20basic.pdf Pristupano: [11.06.2023]

[18] „Stratis Platform and Why it is the Best Blockchain Platform?“ (2021). [Na internetu] Preuzeto: https://www.linkedin.com/pulse/stratis-platform-why-best-blockchain-satya-karki?trk=public_profile_article_view Pristupano [07.09.2023.]

[19] „What are smart contracts and what are they used for?“ (2021). [Na internetu] Preuzeto: https://forkast.news/what-are-smart-contracts-what-are-they-used-for/ Pristupano [15.08.2023]

[20] „What is a Smart Contract How Does it Work?“ (2023). [Na internetu] Preuzeto: https://www.simplilearn.com/tutorials/blockchain-tutorial/what-is-smart-contract Pristupano: [13.08.2023.]

9. Popis slika

10. Popis tablica