SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 7238

ULANČANI BLOKOVI S OGRANIČENIM PRAVOM PRISTUPA I PAMETNIM UGOVORIMA

Filip Anđel

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 7238

ULANČANI BLOKOVI S OGRANIČENIM PRAVOM PRISTUPA I PAMETNIM UGOVORIMA

Filip Anđel

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 7238

Pristupnik:	**Filip Anđel (0036507792)**
Studij:	Računarstvo
Modul:	Računarska znanost
Mentor:	izv. prof. dr. sc. Vladimir Čeperić

Zadatak: **Ulančani blokovi s ograničenim pravom pristupa i pametnim ugovorima**

Opis zadatka:

Opisati osnovnu strukturu, način rada te vrste ulančanih blokova (engl. blockchain) i pametnih ugovora (engl. smart contracts). Napraviti pregled robusnih, skalabilnih ulančanih blokova s ograničenim pravom pristupa i mogućnosti pametnih ugovora, za poslovne primjene uz prikaz njihovih prednosti i nedostataka. Na osnovi pregleda, izabrati jedno od programskih rješenja. Napraviti programsko rješenje tokena putem pametnih ugovora na ulančanim blokovima. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna pojašnjenja i korištenu literaturu.

Rok za predaju rada: 11. lipnja 2021.

Sadržaj

Uvod

Koncept ulančanih blokova (eng. *blockchain*) jedno je od najmlađih područja računarstva, i svakako jedno od zanimljivijih. Potencijalne koristi i prednosti ulančanih blokova su mnogobrojne, a neke možda još nismo ni otkrili. Najpoznatija i prosječnom čovjeku najbliža primjena su svakako kriptovalute, o kojima čujemo svakim danom sve više i više. Temeljne značajke kojima se ulančani blokovi vode jesu decentralizacija, sigurnost, distribuiranost i nepristranost, pa ne začuđuje činjenica da su brzo iskorišteni kao platforma za digitalne valute. Međutim, to je samo jedna od mnogih njihovih primjena, a neke od tih alternativnih primjena ujedno su i tema ovog rada.

Kriptografi su još 80-ih godina prošlog stoljeća dotaknuli koncept ulančanih blokova, no to su bili sami začeci i nikakvo konkretno rješenje nije bilo predloženo. Prva prava implementacija ulančanih blokova dolazi tek 2009., kada anonimna osoba ili grupa ljudi pod pseudonimom Satoshi Nakamoto izdaje Bitcoin, prvi i najpoznatiji *blockchain* (Nakamoto, 2008) na kojem se temelji istoimena kriptovaluta koja je u međuvremenu eksplodirala kako po popularnosti, tako i po vrijednosti.

Nastanak Bitcoina mnogim je inženjerima otkrio cijeli jedan novi svijet ulančanih blokova, te ubrzo počinje rad na razvoju alternativnih ulančanih blokova. Najpoznatiji *blockchain* nakon Bitcoina svakako je tzv. Ethereum, koji 2013. predlaže mladi programer ruskog podrijetla Vitalik Buterin, a uspješno ga implementira 2015. godine (Locke, 2021). Platforma Ethereum osim svoje kriptovalute (ether, **ETH** ili **Ξ)** nudi funkcionalnost pametnih ugovora (*smart contracts*), distribuiranih aplikacija, decentraliziranih financija te NFT-ova. Ovo je znatan pomak u odnosu na Bitcoin koji ne nudi gotovo ništa drugo osim funkcionalnosti kriptovalute.

Međutim, i Bitcoin i Ethereum su javni ulančani blokovi, što znači da je u svakom trenutku bilo komu tko ima pristup Internetu moguće provjeriti stanje *blockchaina* te se njime koristiti. Takvi ulančani blokovi „žele“ da što više ljudi sudjeluje u njima jer na taj način ostvaruju decentralizaciju. To također podrazumijeva da su sve transakcije javne i svima vidljive. No što ako neka poslovna organizacija smatra da bi primjenom ulančanih blokova mogla optimizirati neki dio svojeg poslovnog sustava, a ne želi da njezini podaci i

transakcije budu javno dostupni? Tu u igru ulaze privatni ulančani blokovi s ograničenim pravom pristupa kojima se bavi ovaj rad.

Najpoznatiji primjeri takvih ulančanih blokova su Linuxov Hyperledger Fabric, ConsenSys-ov Quorum i Corda, u vlasništvu tvrtke R3 (Nextrope, 2021). O pojedinostima, prednostima i nedostacima tih platformi govorit ćemo nešto kasnije, a predstavljeno će biti i programsko rješenje na jednoj od navedenih platformi.

1. Ulančani blokovi

Za početak, trebalo bi definirati što je to uopće *blockchain* – ulančani blokovi[1]. U suštini, to nije ništa drugo nego decentralizirana baza podataka, koja umjesto da je locirana na nekom poslužitelju u vlasništvu neke firme ili osobe, ustvari „pripada“ svim njenim korisnicima i nikome od njih u isto vrijeme. Konkretno, to znači da niti jedan korisnik ne može samostalno manipulirati podacima spremljenim na *blockchain* bez dopuštenja i potvrde ostalih korisnika, što sprječava zloupotrebu. Kako se ta decentraliziranost i provjera podataka postiže, bit će jasnije kada shvatimo samu strukturu ulančanih blokova, što uopće sadrže ti blokovi, te kako su povezani u lanac. U sljedećih nekoliko poglavlja bavit ćemo se tim konceptima.

Prije nego što krenemo, bitno je objasniti pojam glavne knjige (eng. *ledger*). *Ledger* služi kao virtualna „knjiga“, umjesto fizičke u tradicionalnom računovodstvu, u koju se zapisuju sve transakcije od početnog bloka pa sve do trenutačnog. Pomoću *ledgera*, kada se sumiraju sve dosadašnje transkacije, dolazimo do trenutačnoga stanja na *blockchainu*.

Blockchain je iznimno opširna tema, te je za razumijevanje ovog rada bitno znati da nisu svi dijelovi teksta ključni. Na mnogim mjestima ubačeni su primjeri i koncepti koji nisu od esencijalne važnosti za osnovnu definiciju *blockchaina*, te samim time neće biti detaljno objašenjeni, već služe da bolje dočaraju i objasne zašto je *blockchain* napravljen takav kakav je, s kojim se problemima susreće i slično. U konačnici, na taj je način lakše razumjeti motivaciju i razloge iza njegovih osnovnih principa.

1.1. Struktura ulančanih blokova

Kako bismo ostvarili koncept ulančanih blokova, potrebni elementi pojašnjeni u sljedećim poglavljima (Lastovetska, 2021). Za primjere je iskorišten Bitcoin, s obzirom na

[1] Kako je riječ o novijim tehnologijama, hrvatsko nazivlje samo djelomice ide ukorak s engleskim. O terminologiji vidi Babić, 2019. U radu će se u naslovima poglavlja koristiti hrvatske inačice gdje god je to moguće, a u tekstovima će se kombinirati kurzivirano nazivlje u izvornome obliku na engleskome jeziku i hrvatske inačice.

to da je prvi pravi *blockchain*, a ujedno i najveći i najpoznatiji, te služi kao inspiracija svim ostalim *blockchainovima*. Treba imati na umu da sljedeći opis vrijedi specifično za platformu Bitcoin, dok se druge izvedbe mogu razlikovati po nekim elementima. Specifične razlike ostalih vrsta *blockchainova* bit će pojašnjene kasnije.

1.1.1. Čvor

Jedan čvor (eng. *node*) označava jedno računalo koje „sudjeluje“ u *blockchainu.* Svaki takav čvor pohranjuje, dobiva i pohranjuje neovisnu kopiju *ledgera* pri nastajanju. Isto tako, svaki novi blok koji se stvori šalje se svim čvorovima. Zatim, čvorovi provjere informacije zapisane u bloku i ako su validne, dodaju blok na svoj lokalni *blockchain*.

Bitno je razumjeti razliku između čvora i korisnika, jer nije svaki korisnik koji se koristi *blockchainom* ujedno i čvor, iako može biti. Na primjeru Bitcoina, da bi se netko njime koristio dovoljno je stvoriti tzv. digitalni novčanik (eng. *wallet*) pomoću neke od mnogobrojnih aplikacija namijenjenih za to. *Wallet* označava jedinstveni zapis identiteta nekog korisnika, te se sastoji od privatnog ključa i javnog ključa (Frankenfield, Bitcoin Wallet, 2020). Jednom kada korisnik ima svoj *wallet,* on može slati, primati i posjedovati bitcoine[2]. Javni ključ pri tome služi kao adresa na koju korisnik može primiti bitcoin, te s koje drugima može slati svoj bitcoin. Privatni ključ jedinstveno je vezan za javni ključ na način da je posjedovanjem privatnog ključa moguće upravljati javnim, što praktički znači da privatni ključ služi kao svojevrsna zaporka. Svoj privatni ključ korisnik mora čuvati od javnosti, inače mu bilo tko može ukrasti bitcoine.

No da bi korisnik upravljao vlastitim čvorom, potrebno je konstantno imati uključen program Bitcoin Core. Jednom kada neko računalo ima pokrenut Bitcoin Core, možemo ga zvati čvorom i ono sada sudjeluje u procesu validacije blokova u suradnji s kasnije objašnjenim rudarima.

[2] Kada se govori o *blockchain* platformi, koristi se naziv Bitcoin pisan velikim slovom, a kada se misli na *token* na istoimenoj platformi, koristi se bitcoin pisano malim slovom. Što je točno *token* bit će objašnjeno kasnije, a trenutno je dovoljno gledati *token* kao digitalnu valutu.

1.1.2. Transakcija

Transakcija je osnovni element *blockchaina* kojim se provodi neka promjena na *ledgeru*. Na primjer, jedno slanje bitcoina drugom korisniku bila bi jedna transakcija. Sve transakcije poslane na *blockchain* pohranjuju se u blokove te ih se onda procesima rudarenja i validacije potvrđuje.

Primjerice, transakcija koja ne bi bila dozvoljena je kada bi korisnik s *walleta* na kojem se nalazi 0.5 bitcoina probao poslati 2 bitcoina na drugi *wallet*. S obzirom na to da transakcija pokušava poslati više bitcoina nego što se nalazi na *walletu*, ona ne smije biti provedena, a o tome se brinu čvorovi i rudari, posebni čvorovi koji sudjeluju u procesu validacije blokova.

1.1.3. Blok

Blok je struktura podataka u kojoj se pohranjuje skup transakcija. Blok se dijeli svim čvorovima u mreži. Svaki blok sadrži kriptografski zapis (eng. *hash*) bloka koji mu prethodi, te se na taj način blokovi povezuju u lanac. Retroaktivna izmjena nekog prošlog bloka nije moguća a da se svaki blok nakon toga također ne izmijeni (jer se mijenjanjem bloka promijeni i njegov *hash*) čime se osigurava da nije moguće zlonamjerno dodati neku lažnu transakciju koja se nije stvarno dogodila.

Svaki se blok sastoji od sljedećih dijelova:

- neki podaci;
- *hash* prethodnog bloka;
- *hash* ovog bloka.

Ovisno o *blockchainu*, podaci spremljeni u blok mogu biti različiti. Bitcoin primjerice pohranjuje podatke o pošiljatelju, primatelju i količini Bitcoina koja se šalje. Blokovi Bitcoina ograničeni su na 1 MB (megabajt). To je ograničenje postavio Satoshi Nakamoto pri stvaranju Bitcoina, a nazivamo ga *block size* s obzirom da ustvari određuje maksimalnu veličinu koju jedan blok smije zauzeti. Sve transakcije koje ne stanu u taj 1 megabajt bit će validirane kasnije, u nekom drugom bloku.

1.1.4. Lanac

Lanac je slijed povezanih blokova, poredan od početnog bloka pa do zadnjeg kronološkim redom. Lanac blokova se često vizualizira kao vertikalan stupac, gdje je početni blok temelj, a svaki se sljedeći blok dodaje na prethodni. Zato se često za određeni blok u lancu kaže visina (eng. *height*). Na primjer, peti blok po redu je na visini 4, jer je početni blok (eng. *genesis block*) na visini 0.

1.1.5. Rudari

Rudari (eng. *miners*) su posebna vrsta čvorova koji „pronalaze“ sljedeći blok pomoću *brute force* metode, što znači da samo isprobavaju razne kombinacije *hasha* sljedećeg bloka sve dok slučajno ne naiđu na pravu (Brock, 2016). Takvi se izrudareni blokovi zatim šalju ostalim čvorovima na validaciju, i ako se uspostavi da je neki pojedini rudar zaista prvi pronašao sljedeći blok, tom se rudaru dodjeljuje nagrada (eng. *mining reward*). Ta nagrada za rudarenje dolazi iz onog dijela bitcoina koji još nije u cirkulaciji, pa rudarenjem praktički stvaramo novi, još neviđeni bitcoin – od čega potječe i naziv *mining*. Ova se metoda potvrđivanja (validacije) blokova naziva „dokaz izvršenog rada“ (eng. *Proof-of-work*), no vidjet ćemo kasnije da neke druge izvedbe *blockchaina* uopće ne koriste rudarenje.

No čemu uopće služi rudarenje? Nije li dovoljno samo skupiti transakcije u blok, poslati ga čvorovima na validaciju, te ako je potvrđen dodati ga na *blockchain*? U praksi se pokazalo da nije, jer u takvoj izvedbi lako dolazi do gubitka decentralizacije. Dovoljno bi bilo da neka skupina ljudi ili organizacija skupi ogroman broj računala i na njima pokrene čvorove. Kada bi uspjeli pod svojom kontrolom posjedovati više od pola svih čvorova u mreži, u teoriji bi mogli nesmetano mijenjati sadržaj blokova odnosno transakcija u svoju korist, takve blokove lažno validirati na svojim čvorovima, te bi tako narušili cijeli smisao *blockchaina* i njegovu sigurnost i konzistentnost. Ta se pojava naziva *51% attack*, a *Proof-of-work* jedan je od načina kako ju spriječiti (Digital Currency Initiative, 2019).

Uvođenjem rudarenja, u sustav se uvodi hardversko ograničenje – sada umjesto da se jedno računalo broji kao jedan čvor, potrebno je imati dovoljnu brzinu pronalaženja sljedećeg bloka, koja se mjeri u *hashevima* po sekundi, ili H/s. Rudarenje je ustvari umjetna prepreka stavljena u sustav kako bi se smanjila vjerojatnost da se dogodi *51% attack*, jer je potrebno imati ogroman *hashrate* pod svojom kontrolom da bi netko preuzeo

blockchain. *Hashrate* se postiže korištenjem specifične opreme za rudarenje koja je relativno skupa. U vrijeme pisanje ovog rada ukupni *hashrate* Bitcoina iznosi oko 150 milijuna TH/s (milijardi *hasheva* po sekundi) (YCharts Inc., 2021), tako da bi potencijalni napadač trebao nekako dovesti još 150 milijuna TH/s u sustav, a za to bi morao kupiti ogromne količine opreme za rudarenje, količinu koju si praktički nitko na svijetu ne može priuštiti. Ovime se smanjuje vjerojatnost napada i preuzimanja *blockchaina*. Naravno, sva ta oprema koristi velike količine električne energiju za rudarenje, pa su *blockchainovi* koji koriste *Proof-of-work* metodu često kritizirani kao loši za okoliš i energiju (Criddle, 2021).

1.1.6. Konsenzus

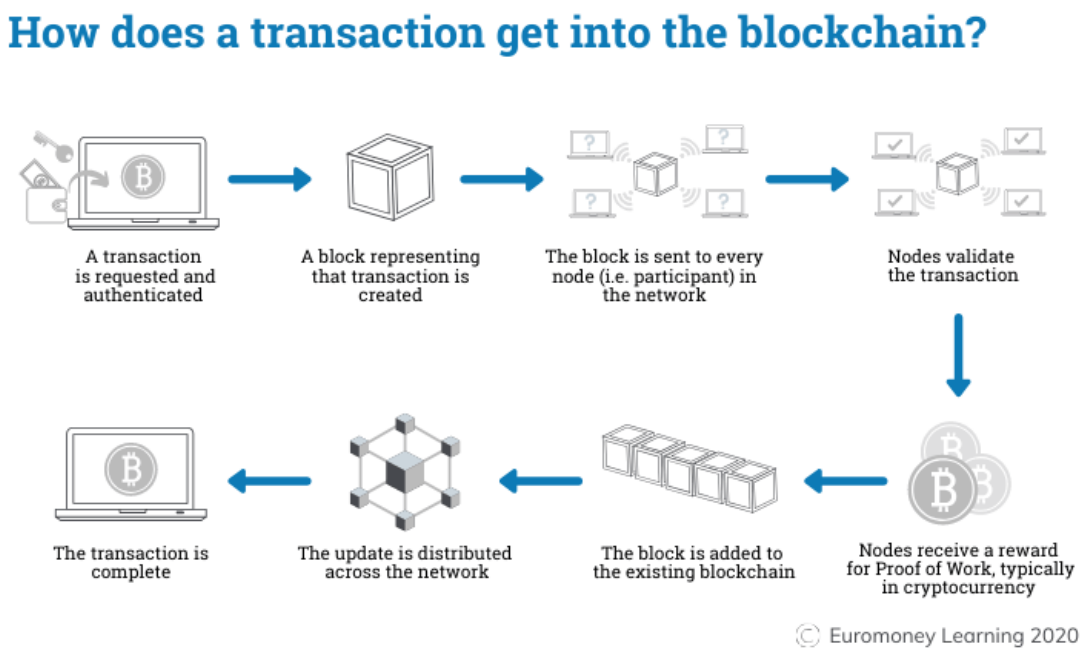
Konsenzus (eng. *consensus protocol*) je skup pravila i uvjeta koji određuju kako se obavljaju operacije na *blockchainu*. Da bi neki čvor sudjelovao u *blockchain* mreži, on mora prihvatiti i primijeniti prethodno dogovorena pravila konsenzusa. Na primjer, jedno od Bitcoinovih pravila konsenzusa je da nikad neće postojati više od 21 milijun bitcoina, te da se nagrada za verifikaciju bloka (rudarenjem) prepolavlja svakih 200 tisuća blokova. Za vrijeme pisanja ovog rada, oko 18.5 milijuna bitcoina je već izrudareno, što ostavlja samo oko 2.5 milijuna koji još nisu u cirkulaciji. Trenutna nagrada za rudarenje bloka iznosi 6.25 bitcoina, a sljedeće se prepolavljanje (*halving*) predviđa za 2024. godinu (Conway, 2021).

Prethodno spomenuti *block size* od 1 MB također je dio konsenzusa. U praksi se pokazalo da je ovo ograničenje prestrogo, odnosno da blokovi nisu dovoljno veliki da bi Bitcoin ostao brz i efikasan kada ga koristi veći broj ljudi, jer u njih ne stane dovoljno transakcija. Postojalo je više pokušaja da se veličina blokova promijeni, međutim, to nije tako jednostavno kako zvuči.

Kao što je već spomenuto, svi se čvorovi jednog *blockchaina* moraju složiti oko konsenzusa kako bi sudjelovali u *blockchainu*. U slučaju da se dogovori promjena konsenzusa, te da svi čvorovi jednoglasno prihvate promjenu, odnosno da se svaki čvor na svijetu složi da se konzensus promijeni, onda bi tranzicija prošla glatko. No u praksi to nikada nije slučaj, jer je dovoljno da se nekolicina čvorova ne složi oko promjene konsenzusa pa da se tranzicija ne obavi u cijelosti. U tom slučaju dolazi do *hard forka,* odnosno podjele jednog *blockchaina* na dva nova – jedan sa starim protokolom konsenzusa, a drugi s novim. U tom slučaju, čvorovi koji nisu prihvatili promjenu konsenzusa dalje vrše validaciju blokova po starom konsenzusu, a čvorovi koji promjenu jesu prihvatili validiraju blokove po novom konsenzusu. Tako se praktički jedan

blockchain dijeli na dva, od kojih je jedan original a drugi nazivamo *fork* (Peters, 2021). Oba *blockchaina* imaju isti početni blok, a lanci izgledaju međusobno identično sve do bloka u kojem se dogodio *fork*. No nakon tog bloka, stari *blockchain* nastavlja dalje po starim pravilima, dok novi stvara potpuno drukčije, nove blokove.

Primjeri takvih *forkova* u slučaju Bitcoina su Bitcoin Gold, Litecoin, Bitcoin XT, Bitcoin Cash i još mnogi drugi. Bitcoin je, u potrazi za rješenjem za spore i neefikasne transakcije, mnogo puta *forkan* u razne druge *blockchainove*. Međutim, niti jedan *fork* Bitcoina zasad nije uspio preuzeti pravi, originalni Bitcoin po popularnosti, iako mu se Bitcoin Cash poprilično približio.



Slika 1. Tijek provođenja jedne transakcije (Euromoney, 2020)

2. Vrste ulančanih blokova

Blockchain se može podijeliti na sljedeće vrste (DataFlair, 2018):

- javni;
- privatni;
- konzorcij;
- hibridni.

2.1. Javni ulančani blokovi

Prethodno spomenuti Bitcoin i Ethereum primjeri su javnih *blockchainova*. Sve kriptovalute dostupne na tržištu također su, po definiciji, javni *blockchainovi.* Javni *blockchain* također je poznat kao *permissionless blockchain*, gdje se naziv odnosi na činjenicu da nije potrebna nikakva posebna dozvola da bi se pristupilo *blockchainu.*

2.1.1. Karakteristike javnih ulančanih blokova

Javni *blockchain* u potpunosti je decentraliziran, što znači da ne postoji nikakvo centralno tijelo koje donosi odluke. Bilo koja osoba bilo gdje u svijetu, pod uvjetom da ima pristup internetu, može se njime koristiti, provjeriti stanje *ledgera*, slati i primati neki token i validirati blokove. Dovoljno je preuzeti neku aplikaciju koja upravlja *walletom* za odabrani *blockchain*, ili u slučaju da korisnik želi sudjelovati u procesu validacije blokova, preuzeti i pokrenuti *miner*, odnosno program koji obavlja rudarenje (naravno pod uvjetom da *blockchain* koristi rudarenje za validaciju blokova).

Javni *blockchain* je transparentan, što znači da su sve transakcije javno vidljive i jasno prikazane. Na primjer, na Bitcoin *blockchainu* moguće je pogledati adresu *walleta* pošiljatelja i primatelja te količinu poslanog Bitcoina za bilo koju transakciju, što do neke razine smanjuje anonimnost. Kada bismo znali adresu *walleta* neke osobe, znali bismo točno koje transakcije je ta osoba obavila i kada.

Naposljetku, javni *blockchain* ima svojstvo nepromjenjivosti. Transakcija koja je jednom poslana na *blockchain* i potvrđena ne može se više nikada promijeniti.

2.1.2. Prednosti javnih ulančanih blokova

Kao što je već spomenuto, javni je *blockchain* dostupan svima, što je bitno ako želimo ostvariti decentraliziranu valutu kao što je Bitcoin. Kada bi postojala neka ograničenja tko može pristupiti *blockchainu*, valuta bi izgubila smisao.

Sljedeće prednosti su sigurnost i povjerenje. Karakteristike decentralizacije, nepromjenjivosti i transparentnosti omogućuju potpuno povjerenje korisnika, jer postoji potpuna sigurnost od zloupotrebe ili zlonamjerne manipulacije podacima. Drugim riječima, u slučaju Bitcoina, ne postoji način da se Bitcoini koji pripadaju nečijem *walletu* oduzmu, što nije slučaj kod centraliziranih financijskih rješenja. Na primjer, poslužitelj koji banka koristi za pohranjivanje podataka o bankovnim računima svojih klijenata teoretski može biti hakiran i njihovi novci ukradeni, što je kod javnih *blockchainova* fizički nemoguće. Naravno, kod *blockchaina* uvijek ostaje mogućnost ljudske pogreške – na primjer, Bitcoin slučajno poslan na krivu adresu izgubljen je zauvijek.

2.1.3. Nedostaci javnih ulančanog blokova

Javni *blockchain* za verifikaciju transakcija koristi se jednom od dviju metoda: prethodno spomenuti dokaz izvršenog rada, ili dokaz uloga (eng. *Proof-of-stake*). Već spomenuti *Proof-of-work*, gdje se verifikacija obavlja rudarenjem, odnosno radom procesorskih jezgri (često se koriste grafičke kartice jer sadrže velik broj jezgri) prilično je neefikasan, i to u više aspekata. Prvi je potrošnja energije koja je uzrokovana velikim brojem računala uključenim u proces rudarenja. Bitcoin primjerice ima godišnju potrošnju električne energije oko 110 TWh, što je otprilike količina energije koju u godinu dana potroši cijela Švedska (Carter, 2021).

Osim toga, javni *blockchain* teško je skalirati na veću količinu korisnika, odnosno transakcija. Kod Bitcoina, koji ima relativno mali *block size*, lako dođe do zagušenja ako se u nekom trenutku pokušava obaviti velik broj transakcija. Bitcoin je namješten tako da se jedan blok izrudari otprilike svakih 10 minuta, a s obzirom na to da je veličina bloka fiksna (1 megabajt) moguće je svakih 10 minuta provesti samo taj jedan megabajt transakcija. Veličina pojedine transakcije varira, tako da je u jedan blok moguće pohraniti oko tisuću transakcija, ali može ih biti i znatno manje. Prosječna brzina za Bitcoin iznosi 4.6 transakcija po sekundi. (L., 2019)

Zbog ovih problema, javni *blockchaini* teže promjeni u neki efikasniji sustav, kako bi se transakcije provodile brže, jeftinije i efikasnije. Ethereum, primjerice, planira promjenu u *Proof-of-stake* sustav verifikacije koji bi trebao znatno ubrzati transakcije i smanjiti potrošnju energije (Ogino, 2021).

2.2. Privatni ulančani blokovi

Za razliku od javnog, privatni se *blockchain* još naziva i *permissioned blockchain*, odnosno *blockchain* s ograničenim pravom pristupa. To znači da ne može bilo tko postati korisnik *blockchaina,* niti bilo tko može verificirati transakcije na *blockchainu.* Takva primjena ima smisla unutar nekog zatvorenog kruga, kao, na primjer, neka organizacija koja u sklopu nekog poslovnog rješenja treba *blockchain.* Za takvu primjenu nema potrebe da ljudi koji nisu unutar te organizacije vide i sudjeluju u *blockchainu*, štoviše, to bi bilo štetno jer bi sve transakcije bile javno poznate.

2.2.1. Karakteristike privatnih ulančanih blokova

Kao što je već spomenuto, jedna od glavnih karakteristika privatnih *blockchainova* jesu privatne transakcije. Time se postiže mogućnost da se provode transakcije koje bi za korisnika bile štetne da su javno poznate, na primjer, cijene usluga i detalji ugovora koji su često poslovna tajna.

Privatni *blockchain* ima ograničeno pravo pristupa, što znači da mora postojati neki proces autentifikacije korisnika. Točnije, neko centralno tijelo unutar mreže *blockchaina* mora imati mogućnost dodati, odnosno utvrditi identitet drugih korisnika te im tako dopustiti pristup i korištenje mreži. Evidentno je odmah da se dodavanjem takvog procesa gubi decentralizacija.

2.2.2. Prednosti privatnih ulančanih blokova

Privatni *blockchain* u pravilu je brži od javnog, skalabilniji i efikasniji. To proizlazi iz činjenice da nisu potrebni svi čvorovi u sustavu da bi se neka transakcija potvrdila. Na primjer, za potvrdu transfera između dvaju korisnika dovoljno je da oba korisnika potvrde tu transakciju da ona bude provedena, za razliku od javnog *blockchaina* gdje se svaki čvor mora složiti s transakcijom kako bi ona bila validna. Time se sustav znatno ubrzava.

Još jedna prednost je laka promjena konsenzusa *blockchaina*. S obzirom na to da se za korištenje privatnog *blockchaina* potrebno unaprijed dogovoriti između svih budućih korisnika, te korisnike u budućnosti ništa ne sprječava da se dogovore i promijene pravila *blockchaina* koji koriste, u slučaju da im je potrebna neka druga funkcionalnost. S javnim *blockchainom* je to nemoguće a da ne dođe do tzv. *hard forka*, što je prethodno objašnjeno.

2.2.3. Nedostaci privatnog ulančanog bloka

Nedostaci ovakve implementacije *blockchaina* su centralizacija, netransparentnost i gubitak anonimnosti. Uključivanjem nekog centralnog tijela u mrežu gubi se decentraliziranost, ali i anonimnost jer je nemoguće potvrditi identitet anonimnog korisnika. Centralni autoritet za autentifikaciju mora znati komu daje pravo pristupa *blockchainu*, dok kod javnog *blockchaina* to nije tako, pa za stvaranje *walleta* na javnom *blockchainu* nije potrebno dati nikakve osobne podatke. Gubitak anonimnosti unutar privatnog *blockchaina* nije pretjerano bitan s obzirom na to da je njegova namjena da se koristi unutar zatvorenog kruga organizacije čiji se članovi već međusobno „poznaju“.

2.3. Konzorcij i hibridni ulančani blok

Consortium blockchain je podvrsta privatnog *blockchaina* koju koristi više entiteta (organizacija). Umjesto jednog entiteta koji ima više članova kao kod privatnog *blockchaina*, sada postoji više entiteta od kojih svaki može imati svoje članove. U načelu, ta je implementacija vrlo slična privatnom *blockchainu* po karakteristikama i načinu rada, samo omogućuje međusobnu suradnju više entiteta.

Hibridni *blockchain* sadrži elemente privatnog i javnog *blockchaina*. Na njemu se neki dijelovi mogu stvoriti kao privatni, a neki kao javni. Na taj način može se na istom *blockchainu* ostvariti i javni i privatni dio, umjesto da se koriste više odvojenih *blockchainova*.

3. Pametni ugovori

Iz prethodnih primjera vidjeli smo da je Bitcoinov *blockchain* po dizajnu ograničen na jednostavne financijske transakcije – posjedovanje, slanje i primanje Bitcoina. To je i logično, s obzirom na to da je Satoshi Nakamoto zamislio Bitcoin kao novu globalnu digitalnu valutu, koja bi bila decentralizirana, a ne pod kontrolom neke države ili banke.

Međutim, može li *blockchain* biti više nego samo digitalni novac? Odgovor na to pitanje pronalazimo u pojmu 'pametni ugovor' (eng. *smart contract*). Pametne ugovore definirao je ranih 90-ih računarski znanstvenik, kriptograf i pravni učenjak Nick Szabo, koji pod pametnim ugovorom smatra „skup obećanja specifiranih u digitalnom obliku, koji uključuje protokole kojima će ugovornici ispuniti dana obećanja“ (Frankenfield, Smart Contracts, 2021).

Najpoznatiji i jedan od prvih primjera *blockchaina* koji koristi pametne ugovore jest Ethereum, čiji je temeljni blok postavljen 2015. Nastankom Ethereuma, definicija pametnih ugovora malo se mijenja. „Službena“ definicija, koju daje Američki nacionalni institut za standarde i tehnologiju postaje „skup koda i podataka (ponekad zvanih funkcijama i stanjima) koji je primijenjen koristeći kriptografski potpisane transakcije na blockchain mreži" (Yaga, Mell, Roby, & Scarfone, 2018). Takva interpretacija pametnih ugovora, koju sada koriste Ethereum Foundation i IBM, umjesto klasičnog koncepta ugovora podrazumijeva bilo kakav računalni program. Pametni ugovor također se može gledati kao zaštićeni računalni kod čije su izvođenje i posljedice izvođenja (kao npr. transfer neke vrijednosti između korisnika) strogo određeni i pohranjeni na blockchain, što znači da ne mogu biti manipulirani nakon što je provedena transakcija sa specifičnim detaljima ugovora. Osim toga, moguće je postaviti neki specifičan uvjet ili skup uvjeta koji moraju biti ispunjeni kako bi se taj kod mogao izvesti, što ustvari intuitivno i podsjeća na ugovor.

Još jedna promjena koju je Ethereum uveo u *blockchain* ekosustav su *tokeni*. Nastankom Ethereuma, dovoljno je putem pametnog ugovora stvoriti neki *token*, koji je moguće slati, primati, posjedovati, provjeriti stanje i slično, kako bi stvorili posve novu kriptovalutu ili *token* za neku drugu svrhu. Zapravo, nije više potrebno za svaki novi *token* stvarati cijeli novi *blockchain*, nego je sada moguće stvoriti *token* na postojećoj Ethereum

platformi, što znatno smanjuje vrijeme i trud potreban za stvaranje *tokena*. Primjeri takvih *tokena* su mnogobrojni, a najpoznatiji je svakako Tether, *token* koji reprezentira valutu USD (američki dolar) i čija je vrijednost u svakom trenutku vezana za vrijednost dolara putem pametnih ugovora. Još jedan primjer zanimljiv za promotriti je Uniswap, također na Ethereum platformi. Uniswap nije samo *token* nego protokol provođenja pametnih ugovora koji ostvaruje svojevrsnu decentraliziranu burzu (eng. *decentralized exchange* ili *DEX*), gdje je moguće zamijeniti bilo koji *token* za neki drugi *token* (naravno pod uvjetom da su oba *tokena* na Ethereum platformi) (Adams, Zinsmeister, & Robinson, 2020). Iz ovih primjera vidimo da je na jednom *blockchainu* moguće ostvariti mnogo različitih funkcija, pa čak i međusobnu interakciju *tokena*, a sve pomoću pametnih ugovora. Još neki zanimljivi primjeri mogućnosti pametnih ugovora su Polygon, Chainlink, Basic Attention Token, 0x i mnogi drugi.

Naposljetku, bitno je spomenuti i Ethereumov *native token*, odnosno ether, također ostvaren pomoću pametnih ugovora. Kada se govori o Ethereumu u kontekstu kriptovalute, zapravo se misli na ether, jer je nemoguće „kupiti“ *blockchain*, može se kupiti jedino *token* na *blockchainu*. Primjerice, u trenutku pisanja ovog rada jedan ether moguće je kupiti za 2705.82 dolara, odnosno 16688,42 kuna (CoinMarketCap, 2021). Pametnim ugovorima postavljeno je pravilo da ethera može postojati maksimalno 116,157,349, što trenutnu tržišnu kapitalizaciju ethera postavlja na 314 milijardi dolara. Kada bi tu tržišnu kapitalizaciju usporedili s kapitalizacijom dionica, ether bi bio u rangu s dionicama tvrtki kao što su The Walt Disney Company ili PayPal (CoinMarketCap, 2021).

4. Nedjeljivi tokeni

Jedna od prvih posljedica nastanka pametnih ugovora je pojava NFT-ova odnosno *non-fungible tokena*. Kako bismo bolje razumjeli o čemu je riječ, krenimo prvo od „običnih“ djeljivih tokena kao što su bitcoin, ether, tether, litecoin... U pravilu možemo reći da je svaki djeljivi (eng. *fungible*) token implementiran na javnom *blockchainu* jedna kriptovaluta (Frankenfield, Crypto Tokens, 2020). S obzirom na to da se radi o valuti, sve mogućnosti i karakteristike djeljivih tokena u skladu su s intuitivnim mogućnostima i karakteristikama običnih *fiat* valuta kao što je hrvatska kuna (HRK) ili američki dolar (USD).

Prethodno spomenuti *tokeni* korišteni kao kriptovalute ostvareni su na način da je svaki token jednako vrijedan i ravnopravan drugom tokenu i može biti podijeljen do neke unaprijed određene decimale. Bitcoin primjerice ide do 8 decimalnih mjesta, što znači da je najmanja količina bitcoina koju je moguće posjedovati, slati itd. (iako slanje takvih malih količina u praksi nije preporučljivo) 0.00000001 BTC, također poznato kao jedan *satoshi*. Kod djeljivih tokena, 1 bitcoin je jednako 1 bitcoin, bez obzira na to tko je i kada te Bitcoine izrudario, kome ih je slao, i tako dalje – u potpunosti su međusobno jednaki i zamjenjivi, kao što su u stvarnosti dvije novčanice od 100 hrvatskih kuna međusobno ekvivalentne, bez obzira na to kada i gdje su tiskane, tko ih je posjedovao i slično.

NFT je upravo suprotno. NFT označava token koji je jedinstven po nečemu (najčešće im se daje jedinstveni identifikacijski broj), te sadrži jedinstveni potpis koji označava da je jedan takav NFT originalan (Sharma, 2021). Jedan NFT nije moguće samo zamijeniti drugim, jer je svaki NFT jedinstven. Osim identifikacijskog broja i potpisa, standardna implementacija NFT-a podrazumijeva još i jedan „odlomak“ proizvoljno odabranih podataka koji se mogu „pohraniti“ u jedan NFT. Primjerice, najpoznatija primjena NFT-eva jest digitalna umjetnost. To funkcionira na sljedećem principu: umjetnici stvore neko umjetničko djelo u obliku računalno generirane slike te ga pohrane u NFT. Tim činom taj NFT postaje originalni zapis tog umjetničkog djela, a u potpisu NFT-a sadržan je i zapis o vlasniku. Time smo postigli sustav sličan klasičnim umjetninama – iako bilo tko može fotografirati neko umjetničko djelo i imati vlastitu kopiju za sebe, samo je jedan original. Analogno, moguće je na internetu pogledati razne digitalne umjetnine u obliku NFT-a, pa

čak ih i kopirati, ali ako želimo posjedovati original, potrebno je kontaktirati vlasnika i kupiti baš taj specifični originalni NFT. Na ovaj način NFT-evi postižu kolekcionarsku vrijednost, iako je takva primjena samo jedna od mnogih mogućih.

Jedna od teoretskih primjena NFT-ova mogla bi biti certifikat vlasništva (Farooqui, 2021). Primjerice, vlasništvo auta – u trenutačnoj izvedbi, neka baza podataka u vlasništvu države pohranjuje podatke o pojedinom autu i njegovu odgovarajućem vlasniku. U takvom centraliziranom sustavu moguće je zlonamjerno ili greškom krivo izmijeniti nečiji certifikat. Osim toga, ako poslužitelj na kojem se nalazi baza podataka iz nekog razloga prestane raditi, nemoguće je provjeriti vlasništvo auta dok ga se ponovno ne osposobi. Kada bi se certifikat o vlasništvu auta (to se može primijeniti i na nekretnine i mnoge druge oblike imovine) pohranio na *blockchain* u obliku NFT-a, on bi bio zaštićen od bilo kakvih pokušaja zlonamjerne manipulacije, te bi se moglo transparentno i jasno provjeriti prijašnje vlasnike uvidom u *blockchain* i podaci se ne bi izgubili ako neki od čvorova prestane raditi, jer i svi ostali čvorovi imaju svoju vlastitu kopiju *blockchaina*. Takav sustav trenutačno još uvijek nije primijenjen nigdje u svijetu.

5. Privatni ulančani blokovi za poslovnu primjenu

Ubrzo nakon nastanka Ethereuma, prve se tvrtke interesiraju za stvaranje svojeg privatnog *blockchaina* s mogućnosti pametnih ugovora za poslovnu primjenu. Među poznatijim primjerima su već spomenuti Hyperledger, Quorum i Corda.

5.1. Hyperledger

Hyperledger je projekt u vlasništvu zaklade Linux. 2015. godine su najavili stvaranje *blockchaina*, a izdali su ga u siječnju 2018. pod imenom Hyperledger Fabric u suradnji s IBM-om. Hyperledger Fabric je osmišljen kao *blockchain* infrastruktura s ograničenim pravom pristupa s mogućnosti pametnih ugovora, modularnom arhitekturom i učlanjivanjem (Kuhrt, 2021). C*haincode* (naziv koji Hyperledger koristi za pametne ugovore) može biti programiran u bilo kojem od sljedeća tri jezika: Java, JavaScript i Go.

5.2. Quorum

Quorum je *blockchain* platforma s mogućnošću pametnih ugovora (Nelson, 2021). Temelji se na Ethereumu, točnije *fork* je go-ethereuma, što je službena implementacija Ethereuma u programskom jeziku Go. Glavna promjena u odnosu na Ethereum je to što je Quorum privatan *blockchain*. U vlasništvu je tvrtke ConsenSys, koji je Quorum kupio od JPMorgan Chase-a. Kao *fork* Ethereuma, i Quorum koristi Ethereumov jedinstveni *blockchain* jezik Solidity.

5.3. Corda

Platforma Corda proizvod je tvrtke R3 koja se bavi poslovnim primjenama *blockchaina* i razvojem softvera. Corda je *open source* tehnologija bazirana na *blockchainu*, koja nastoji ostvariti distribuirani *ledger*, a ima i mogućnost pametnih ugovora (R3, 2021). Dizajnirana je za pohranu, upravljanje i automatiziranje zakonskih ugovora između poslovnih partnera. Corda je programirana u programskom jeziku Kotlin.

5.4. Usporedba

Što se tiče samog funkcioniranja i načina rada *blockchaina*, sva su tri rješenja dosta slična. Sva tri su privatni *blockchainovi* s mogućnošću pametnih ugovora, a namijenjeni su za poslovnu primjenu. Razlikuju se, primjerice, po algoritmu kojim se postiže konsenzus: kod Hyperledgera konsenzus se može postići ili tako da svi čvorovi u mreži potvrde transakciju, ili da ju potvrde samo članovi koji sudjeluju u transakciji. Quorum i Corda oba zahtijevaju da se samo članovi koji sudjeluju u transakciji dogovore. Hyperledger koristi algoritme KAFKA ili RAFT, Quorum koristi QuorumChain, RAFT ili BFT, a Corda koristi validaciju od strane čvorova koje u svojem sustavu naziva Notaries. Svi od ovih algoritama konsenzusa omogućuju brze, efikasne i lako skalabilne transakcije. Kao što je prethodno spomenuto, privatni *blockchain* puno je lakše skalirati na veći broj korisnika nego javni, pa tako vrijedi i za ove tri platforme. (Blockchain Simplified, 2020)

Od ovih triju, Hyperledger se pokazao najmodularnijim, što znači da je moguće koristiti samo određene module odnosno dijelove platforme koji nam trebaju. Time se smanjuje količina zauzetih resursa koja bi inače otpadala na neiskorištene dijelove koda.

Neke su razlike u načinu autentifikacije čvorova. Kada se novi čvor želi priključiti u mrežu, moraju postojati već unaprijed odabrani čvorovi koji imaju dopuštenje dodati ostale čvorove. Alternativa tomu je da postoji neko centralno tijelo koje odlučuje o tome koji čvor se smije priključiti mreži. Čvorovima je potrebno dodijeliti nekakav identitet, odnosno neku jedinstvenu oznaku prema kojoj možemo sa sigurnošću zaključiti identitet pojedinog čvora. Hyperledger i Corda za identitet koriste PKI (*public key infrastructure)* s malom razlikom u tome da Hyperledger za čvorove nudi jedino identitet na razini organizacije, dok Corda može dati identitet individualnom čvoru neovisno o organizaciji, ili pak cijeloj organiziciji pa je po tome Quorum malo fleksibilniji. Čvorovi u Quorumu dobivaju identitet prema paru javnih i privatnih ključeva.

Iako su platforme dosta slične po funkcijama, karakteristikama i primjenama, u svrhu ovog rada bilo je potrebno izabrati jednu na kojoj bi se izradilo programsko rješenje. Corda i Quorum najčešće se koriste za implementaciju nekih financijskih rješenja, dok je Hyperledger korišten za općenitu primjenu na razini jedne organizacije ili skupa

organizacija, na primjer, za neka logistička rješenja. S obzirom na jezike u kojima su pisane platforme i dostupnost dokumentacije daleko je najpristupačniji Hyperledger, s obzirom da podržava Javu, JavaScript i Go, tri dosta poznata jezika. Iz navedenih je razloga za programsko rješenje izabran Hyperledger.

5.5. Hyperledger Fabric

Hyperledger Fabric platforma je Hyperledgerova implementacija privatnog *blockchaina* s mogućnošću pametnih ugovora za poslovne primjene (Hyperledger, 2020). Postoje mnoga odstupanja od opisanog načina rada *blockchaina* iz prvog poglavlja, a prva i najočitija je što se radi o privatnom *blockchainu*. Dakle, nekoj Hyperledger mreži ne može pristupiti netko tko nije direktno pozvan u nju. Fabric je zamišljen kao *blockchain* mreža koju koristi jedna ili više organizacija. Pod organizacijom se, s obzirom da je sustav zamišljen za poslovnu primjenu, smatra jedna tvrtka. Fabric dopušta stvaranje organizacija, koja dobiva svoj jedinstveni identifikacijski ključ.

U svaku organizaciju mogu se učlaniti *peerovi*, Hyperledgerov naziv za *node*. Svaki takav *peer* pri učlanjivanju predstavlja digitalni certifikat organizacije kojoj pripada te tako postaje dio nje. *Peerovi* mogu pristupiti *blockchainu,* provjeriti stanje *ledgera* i potvrđivati transakcije. Umjesto klasičnog sustava rudarenja, kod Hyperledgera se za validaciju blokova koristi konsenzus između *peerova* što znači da transakcije potvrđuju samo oni *peerovi* kojih se transakcije tiče. Na primjer, za prijenos nekog *tokena* između dvije organizacije potrebna je suglasnost samo *peerova* koji pripadaju tim dvjema organizacijama, što omogućava veliku skalabilnost, brzinu i efikasnost sustava. Naravno, kao što je već poznato za privatne *blockchainove*, takvim se dizajnom gubi decentralizacija.

Hyperledger koristi posebnu vrstu *peera* zvanu *orderer* koji obavlja nizanje transakcija u blokove, koje zatim prosljeđuje *peerovima*. *Orderer* je *peer* zaslužan za održavanje *blockchaina* a osim toga ima i centralnu ulogu u komunikaciji *peerova* i *blockchaina*.

Pametni ugovori koje Hyperledger omogućuje nazivaju se *chaincode*. *Chaincode* je neki softver koji definira *asset* (neki oblik imovine), a zatim i upute za modifikaciju postojećih *assetova* na *ledgeru* u obliku transakcija.

Zaključak

Hyperledger Fabric platforma je korištena za poslovnu primjenu decentralizirane glavne knjige (*ledgera*). Od navedenih robusnih skalabilnih *blockchaina* s ograničenim pravom pristupa i mogućnosti pametnih ugovora, Hyperledger Fabric je najaktivniji, a zajednica koja sudjeluje u njegovom razvoju sve je veća. Mogućnosti koje nudi primjenjive su u područjima kao što su državna vlast, financije, zdravstvo, logistika i još mnoge druge.

Jednostavan primjer takve primjene ostvaren je u programskom rješenju ovog rada. Izrađena je Java aplikacija koja služi kao grafičko sučelje za interakciju s *blockchainom*. Korištena je Hyperledgerova vlastita implementacija jednostavnog NFT-a dana u njihovim primjerima, pod imenom *Secured asset transfer*. Ostvareno je stvaranje jedinstvenih, nedjeljivih *asseta* s određenim svojstvima, mijenjanje njihovih svojstava, čitanje njihovih svojstava, ponuda za kupnju/prodaju nekog *asseta* te konačno prebacivanje *asseta* s jednog vlasnika na drugog.

Osim osnovnog načina funkcioniranja Hyperledger platforme, kroz programsko rješenje moguće je shvatiti i princip rada Fabric Gateway SDK-a za Javu. To je API koji omogućava jednostavnu i intuitivnu komunikaciju između Java aplikacija i Hyperledger Fabric *blockchaina*. Postoje i druge verzije API-jeva za druge programske jezike.

Potencijalne nadogradnje postojeće aplikacije su implementacija preostalih neiskorištenih metoda iz pametnog ugovora *Secured asset transfer*, kao primjerice uvid u povijest prodaje i kupnje nekog *asseta*. Osim toga, neiskorištenih metoda više i nema pa bi za neku kompleksniju aplikaciju bilo potrebno napraviti vlastiti složeniji *chaincode* koji bi implementirao NFT ili neku drugu vrstu *asseta*.

Razumijevanje rada Hyperledger *blockchaina* i pripadnog SDK-a potrebno je za razvoj *blockchain* mreža i odgovarajućih aplikacija, što je danas sve traženije područje informacijskog sektora. Potencijalne primjene *blockchain* tehnologije u poslovnom svijetu, a i svijetu općenito, mnogobrojne su te ih svakodnevno otkrivamo još. *Blockchain* je tehnologija koja ima potencijal unaprijediti značajan dio dosadašnjih sustava, a pogotovo u financijskom (kriptovalute) i poslovnom svijetu (platforme kao Hyperledger).

Literatura

[1] National Institute of Standards and Technology, *Smart contract*. Poveznica: https://csrc.nist.gov/glossary/term/Smart_contract; pristupljeno 11. lipnja 2021.

[2] Digital Currency Initiative, *51% attacks*, (2019, listopad). Poveznica: https://dci.mit.edu/51-attacks; pristupljeno 11. lipnja 2021.

[3] YCharts Inc., *Bitcoin Network Hash Rate*, (2021, lipanj). Poveznica: https://ycharts.com/indicators/bitcoin_network_hash_rate; pristupljeno 11. lipnja 2021.

[4] DataFlair, *Types of Blockchains – Decide which one is better for your Investment Needs*, (2018, studeni). Poveznica: https://data-flair.training/blogs/types-of-blockchain; pristupljeno 11. lipnja 2021.

[5] A. Brock, *Bitcoin miners do nothing except resolve conflicts… NOT*, Medium, (2016, lipanj). Poveznica: https://medium.com/@artbrock/hi-jeff-966e662d86fe; pristupljeno 11. lipnja 2021.

[6] N. Carter, *How Much Energy Does Bitcoin Actually Consume*?, HBR, (2021, svibanj). Poveznica: https://hbr.org/2021/05/how-much-energy-does-bitcoin-actually-consume; pristupljeno 11. lipnja 2021.

[7] L. Conway, *Bitcoin Halving*, Investopedia, (2021, svibanj). Poveznica: https://www.investopedia.com/bitcoin-halving-4843769; pristupljeno 11. lipnja 2021.

[8] C. Criddle, *Bitcoin consumes 'more electricity than Argentina'*, BBC, (2021, veljača). Poveznica: https://www.bbc.com/news/technology-56012952; pristupljeno 11. lipnja 2021.

[9] J. Frankenfield, *Smart Contracts*, Investopedia, (2021, svibanj). Poveznica: https://www.investopedia.com/terms/s/smart-contracts.asp; pristupljeno 11. lipnja 2021.

[10] K. L., *The Blockchain Scalability Problem & the Race for Visa-Like Transaction Speed*, Towards Data Science, (2019, siječanj). Poveznica: https://towardsdatascience.com/the-blockchain-scalability-problem-the-race-for-visa-like-transaction-speed-5cce48f9d44; pristupljeno 11. lipnja 2021.

[11] A. Lastovetska, *Blockchain Architecture Basics: Components, Structure, Benefits & Creation*, MLSDev, (2021, siječanj). Poveznica: https://mlsdev.com/blog/156-how-to-build-your-own-blockchain-architecture; pristupljeno 11. lipnja 2021.

[12] T. Locke, *Why Ethereum founder Vitalik Buterin got into crypto*, CNBC, (2021, svibanj). Poveznica: https://www.cnbc.com/2021/05/18/why-ethereum-founder-vitalik-buterin-got-into-crypto-bitcoin.html; pristupljeno 11. lipnja 2021.

[13] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, Bitcoin, (2008, listopad). Poveznica: https://bitcoin.org/bitcoin.pdf; pristupljeno 11. lipnja 2021.

[14] O. Ogino, *PROOF-OF-STAKE (POS)*, Ethereum, (2021, travanj). Poveznica: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/; pristupljeno 11. lipnja 2021.

[15] K. Peters, *A History of Bitcoin Hard Forks*, Investopedia, (2021, svibanj). Poveznica: https://www.investopedia.com/tech/history-bitcoin-hard-forks/; pristupljeno 11. lipnja 2021.

[16] Nextrope, *4 most popular blockchains - analysis and comparison of Ethereum, Hyperledger Fabric, Corda and Quorum*, (2021, travanj). Poveznica: https://nextrope.com/en/4-most-popular-blockchains-analysis-and-comparison-of-ethereum-hyperledger-fabric-corda-and-quorum/; pristupljeno 11. lipnja 2021.

[17] CoinMarketCap, *Ethereum*, (2021, lipanj). Poveznica: https://coinmarketcap.com/currencies/ethereum/; pristupljeno 10. lipnja 2021.

[18] H. Adams, N. Zinsmeister i D. Robinson, *Uniswap v2 Core*, Uniswap, (2020, ožujak). Poveznica: https://uniswap.org/whitepaper.pdf; pristupljeno 11. lipnja 2021.

[19] CoinMarketCap, *Bitcoin vs The Biggest Companies And Assets In The World by Market Cap*, (2021, lipanj). Poveznica: https://coinmarketcap.com/largest-companies/;

pristupljeno 11. lipnja 2021.

[20] J. Frankenfield, *Crypto Tokens*, Investopedia, (2020, lipanj). Poveznica: https://www.investopedia.com/terms/c/crypto-token.asp; pristupljeno 11. lipnja 2021.

[21] R. Sharma, *Non-Fungible Token (NFT) Definition*, (2021, ožujak). Poveznica: https://www.investopedia.com/non-fungible-tokens-nft-5115211; pristupljeno 11. lipnja 2021.

[22] S. Farooqui, *'There's literally no limit': NFTs could soon be used for cars, real estate: Experts*, Winnipeg City News (2021, travanj). Poveznica: https://winnipeg.citynews.ca/2021/04/06/theres-literally-no-limit-nfts-could-soon-be-used-for-cars-real-estate-experts/; pristupljeno 11. lipnja 2021.

[23] D. Yaga, P. Mell, N. Roby i K. Scarfone, *Blockchain Technology Overview*, NIST, (2018, listopad). Poveznica: https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf; pristupljeno 11. lipnja 2021.

[24] T. Kuhrt, *Hyperledger Fabric*, Hyperledger, (2021, lipanj). Poveznica: https://wiki.hyperledger.org/display/fabric/Hyperledger+Fabric; pristupljeno 11. lipnja 2021.

[25] M. Nelson, *What is ConsenSys Quorum*?, (2021, lipanj). Poveznica: https://consensys.net/blog/quorum/what-is-consensys-quorum/; pristupljeno 11. lipnja 2021.

[26] R3, *Corda Platform*, (2021). Poveznica: https://www.r3.com/corda-platform/; pristupljeno 11. lipnja 2021.

[27] Blockchain Simplified, *Hyperledger vs Quorum vs Corda - Which is correct for your business?*, Blockchain Simplified, (2020, rujan). Poveznica: https://blockchainsimplified.com/blog/hyperledger-vs-quorum-vs-corda-which-is-correct-for-your-business/; pristupljeno 11. lipnja 2021.

[28] Hyperledger, *Hyperledger Fabric*, (2020, veljača). Poveznica: https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html#hyperledger-fabric; pristupljeno 11.

lipnja 2021.

[29] J. Frankenfield, *Bitcoin Wallet*, Investopedia (2020, lipanj). Poveznica: https://www.investopedia.com/terms/b/bitcoin-wallet.asp; pristupljeno 11. lipnja 2021.

[30] A. Babić, *Terminologija ulančanih blokova na hrvatskome jeziku*, *Hieronymus 6,* (2019), str. 27-51.

[31] Euromoney, *How transactions get into the blockchain*, (2020). Poveznica: https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain; pristupljeno 11. lipnja 2021.

Sažetak

Ulančani blokovi s ograničenim pravom pristupa i pametnim ugovorima

Opisana je povijest i razvoj *blockchain* tehnologije. Definirana je osnovna struktura i elementi *blockchaina* i objašnjen njegov način rada. Dana je definicija i primjer pametnog ugovora, kao i definicija i primjer *non-fungible tokena*. Nabrojani su tipovi *blockchaina* i njihove karakteristike i primjene. Spomenute su tri konkretne implementacije privatnih *blockchaina* za poslovne primjene, njihove prednosti i nedostaci. Dan je kratki opis jedne od implementacija. Priloženo je programsko rješenje u obliku aplikacije koja omogućuje interakciju s Hyperledger *blockchainom*.

Ključne riječi: *blockchain*, transakcija, pametni ugovori, *ledger*, *node*, ograničeno pravo pristupa, konsenzus, skalabilnost, API, NFT

Summary

Private blockchains with smart contracts

The history and development of blockchain technology is described. The basic structure, elements, and mechanisms behind blockchain are defined and explained. Smart contracts are defined, and some existing examples are given. Non-fungible tokens are defined, and theoretical examples are given. The different types of blockchain are listed, along with their specific characteristics and uses. Three specific implementations of private blockchains for commercial use and their advantages and disadvantages are outlined. A short description is provided for one of the implementations. A programming implementation of an application that interacts with the Hyperledger blockchain is attached.

Keywords: blockchain, transaction, node, smart contract, ledger, permissioned blockchain, consensus, scalability, API, NFT

Privitak

Programsko rješenje

Princip funkcioniranja Hyperledger Fabric mreže moguće je detaljno proučiti u Hyperledger dokumentaciji. Kroz sljedećih par poglavlja bit će dana objašnjenja programskog rješenja, način rada, verzije softvera korištene za rješenje te upute za korištenje. Potpuni kod priložen je uz rad, a posebno bitni dijelovi koda bit će objašnjeni u ovom privitku.

Secured asset transfer

Na repozitoriju Hyperledger Fabrica dani su *Fabric samples* - primjeri nekih potencijalnih pametnih ugovora koje Hyperledger naziva *chaincode*. C*haincode* koji će se koristiti za programsko rješenje je Secured asset transfer, što je praktički implementacija primitivnog NFT-a. Secured asset transfer omogućava stvaranje jedinstvenog tokena, zvanog *asset*, koji ima sljedeće podatke:

- ID (identifikacijska oznaka);
- veličina;
- boja;
- vlasnik (neka od organizacija u mreži);
- opis.

Specifično za ovaj *chaincode* je što je dio podataka o *assetu* skriven. Svima u mreži javno je vidljiv ID *asseta*, vlasnik *asseta* i opis, dok su veličina i boja privatni. Na samom *blockchainu* zapis o veličini i boji kriptografski je zaštićen te ga jedino njegov trenutni vlasnik može pročitati.

Secured asset transfer implementira razne funkcije (metode) koje *peer* može pozvati kako bi stvorio svoj *asset*, pročitao javne podatke o nekom *assetu,* prodao *asset* i tako dalje. Metode korištene u programskom rješenju su sljedeće:

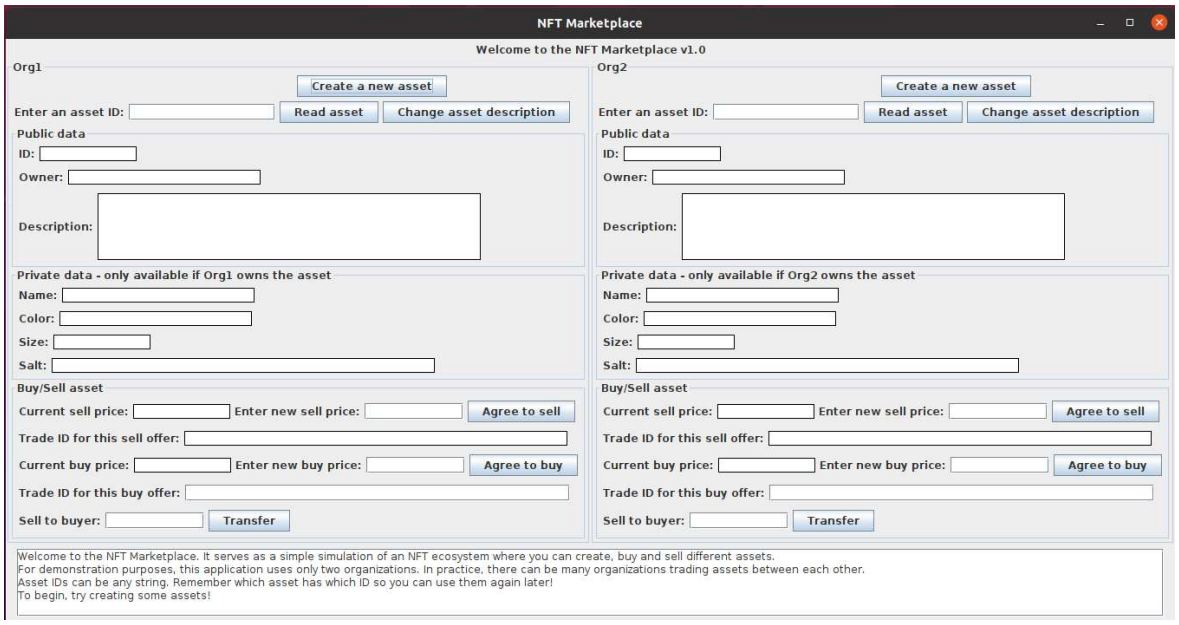
- createAsset;
- readAsset;
- getAssetPrivateProperties;
- changePublicDescription;
- agreeToSell;
- agreeToBuy;
- transferAsset;
- getAssetSalesPrice;
- getAssetBidPrice.

Fabric Gateway SDK for Java

Fabric Gateway SDK je API koji omogućava pisanje Java aplikacija koje se mogu spojiti na *blockchain* mrežu, slati transakcije *blockchainu*, odnosno pozivati određene dijelove *chaincodea* (metode), provjeriti stanje *ledgera* i slično.

NFT Marketplace

Za programsko rješenje ovog rada pomoću Java Swinga napravljen je grafičko korisničko sučelje (eng. *Graphical User Interface* ili *GUI*) koji simulira jednostavnu razmjenu *asseta* na Hyperledger Fabric *blockchainu* između dvije organizacije. Uvidom u kod aplikacije moguće je razumjeti osnove rada s Hyperledger Fabric platformom i kako Fabric Gateway SDK komunicira s *blockchain* mrežom. Aplikacija je nazvana NFT Marketplace, a pisana je u Javi 8, dok je *blockchain* mreža osposobljena pomoću skripti danih u Hyperledgerovim primjerima. Korišteni *chaincode* implementiran je u programskom jeziku Go verzije 1.14. S obzirom da ostvarenje samog grafičkog sučelja nije tema ovog rada, te je Java Swing dobro dokumentirana i poznata biblioteka, nećemo se ovdje baviti tim detaljima aplikacije. Kroz sljedeća poglavlja bit će objašnjen općeniti princip rada aplikacije. Izgled grafičkog sučelja aplikacije vidljiv je na slici 2.



Slika 2. Izgled grafičkog sučelja aplikacije NFT Marketplace

Pokretanje Fabric mreže

Za pokretanje same *blockchain* mreže iskorištene su skripte koje su također uključene u Fabricove primjere, točnije startFabric.sh i networkDown.sh. Te su skripte malo izmijenjene kako bi radile sa Secured asset transfer *chaincodeom*. Prije pokretanja same aplikacije, potrebno je prvo pokrenuti skriptu startFabric kako bi se aplikacija imala na što spojiti. Navedena skripta pokreće simulaciju Hyperledger Fabric mreže, točnije, stvara dvije organizacije (Org1 i Org2) od kojih svaka ima po jednog *peera*. *Peerovi* dobivaju svoje digitalne certifikate koji ih vežu za njihove organizacije. Ti su certifikati dani u obliku *wallet* datoteka. Osim toga, skripta pokreće i *orderer*, te Org1 i Org2 spaja u jedan kanal (eng. *channel)* pod nazivom „mychannel“. Zatim, unutar tog kanala *deploya* Secured asset transfer, odnosno instalira ga na pojedine *peerove* unutar kanala te im tako omogućava da ga koriste i stvaraju transakcije koristeći pametne ugovore definirane u njemu.

EnrollAdmin i RegisterUser

Nakon pokretanja mreže aplikacija može krenuti s radom. Prvi korak koji aplikacija mora obaviti je registrirati se na mrežu, odnosno predstaviti se kao neki *peer*. U ovom slučaju s obzirom da aplikacija služi kao simulacija trgovine između dva lokalno pokrenuta

peera, aplikacija se registrira kao oba u isto vrijeme te će kasnije biti moguće obavljati transakcije kao bilo koji od ta dva *peera*.

Za registraciju kao *peer* korištene su klase EnrollAdmin i RegisterUser. Ove klase su prenamijenjene iz Hyperledgerovih primjera, točnije iz primjera „fabcar“ gdje ispunjavaju istu svrhu kao i ovdje. One učitavaju identitet *peerova* iz datoteke stvorene pri pokretanju mreže, na tom *peeru* stvaraju prvo *Admina*, a zatim *Usera*, te konačno omogućavaju interakciju s Fabric mrežom kao User na prvom ili drugom *peeru*.

```
// enroll admins and register users
// only needs to be ran once per network
EnrollAdmin.main(null);
RegisterUser.main(null);
EnrollAdmin2.main(null);
RegisterUser2.main(null);

// initialize builder of org1
Path walletPath = Paths.get("wallet");
Wallet wallet = Wallets.newFileSystemWallet(walletPath);
// load a CCP
Path networkConfigPath = Paths.get("..", "..", "test-network", "organizations"
        ,"peerOrganizations", "org1.example.com"
        ,"connection-org1.yaml");

Builder builderOrg1 = Gateway.createBuilder();
builderOrg1.identity(wallet, "appUser").networkConfig(networkConfigPath).discovery(true);

// initialize builder of org2
Path walletPath2 = Paths.get("wallet2");
Wallet wallet2 = Wallets.newFileSystemWallet(walletPath2);
// load a CCP
Path networkConfigPath2 = Paths.get("..", "..", "test-network"
        ,"organizations", "peerOrganizations"
        ,"org2.example.com", "connection-org2.yaml");

Builder builderOrg2 = Gateway.createBuilder();
builderOrg2.identity(wallet2, "appUser").networkConfig(networkConfigPath2).discovery(true);



// from this point on, the blockchain can be interacted with
// by using gateways builderOrg1 and builderOrg2
Gateway gateway1 = builderOrg1.connect();
Gateway gateway2 = builderOrg2.connect();

// getting the peers in this channel
Network network = gateway1.getNetwork("mychannel");
Channel channel = network.getChannel();
peers = channel.getPeers();

// initialize GUI
javax.swing.SwingUtilities.invokeLater(new Runnable() {
    public void run() {
        createAndShowGUI(gateway1, gateway2);
    }
});
```

Slika 3. Kod klase main

Glavna metoda

Glavna metoda (*main),* prikazana na Slici 2., se nakon upisivanja *Admina* i registriranja *Usera* može prijaviti kao prethodno napravljeni *User*, a zatim spojiti na *Gateway* prema Fabric mreži. Preko *gatewaya* aplikacija komunicira s *blockchainom* te je moguće koristiti pametne ugovore putem transakcija, provjeravati stanje *ledgera* na pokrenutom *peeru* i slično. Za interakciju s *blockchain* mrežom iz perspektive prve organizacije (Org1) koristi se varijabla „gateway1“, a iz perspektive druge organizacije (Org2) varijabla „gateway2“. U ostatku koda provodit će se razne transakcije kroz koje ostvarujemo interakciju aplikacije s *blockchainom*, a sve će se to raditi pomoću spomenutih *gatewaya*. Nakon toga, u varijablu „peers“ se pohranjuju svi *peerovi* koji su članovi kanala „mychannel“. Točnije to su *peer* iz Org1 i *peer* iz Org2.

Konačno, pokreće se GUI aplikacije.

Stvaranje *asseta*

Pritiskom na dugme u prozoru aplikacije moguće je stvoriti novi *asset*. Otvara se novi prozor u koji je potrebno upisati podatke o *assetu* koji želimo stvoriti. Ti se podaci prenose metodi createAsset, prikazanoj na slici 3.

```
protected static void createAsset(int org, Gateway gateway, String id, String color, String size, String description){
    try {
        Network network = gateway.getNetwork("mychannel");
        Contract contract = network.getContract("secured");

        Transaction createAsset = contract.createTransaction("createAsset");

        JSONObject assetProperties = new JSONObject();
        Field changeMap = assetProperties.getClass().getDeclaredField("map");
        changeMap.setAccessible(true);
        changeMap.set(assetProperties, new LinkedHashMap<>());
        changeMap.setAccessible(false);

        Map<String, byte[]> transientData = new HashMap<String, byte[]>();
        assetProperties.put("object_type","asset_properties");
        assetProperties.put("asset_id", id);
        assetProperties.put("color", color);
        assetProperties.put("size", Integer.parseInt(size));
        assetProperties.put("salt", RandomStringUtils.randomAlphanumeric(40));

        transientData.put("asset_properties", assetProperties.toString().getBytes("utf-8"));

        createAsset.setTransient(transientData);

        createAsset.submit(id, description);

        console.append("Org" + org + " successfully created asset " + id + "\n");
    } catch(Exception e) {
        console.append(e.toString() + "\n");
    }
}
```

Slika 4. Kod metode createAsset

Vidimo da se metodi predaju varijable potrebne za stvaranje novog *asseta*. Prvo što ona radi je dohvaća kanal „mychannel“ a zatim dohvaća pametni ugovor „secured“ (misli se na Secured asset transfer) u varijablu „contract“. Nakon toga, moguće je iskoristiti metodu createTransaction u klasi Contract kako bi stvorili transakciju pod nazivom CreateAsset. Ta transakcija CreateAsset implementirana je u samom kodu pametnog ugovora, a moguće ju je pogledati u datoteci asset_transfer.go u direktoriju „asset-transfer-secured-agreement/chaincode-go“.

Transakcija CreateAsset dio argumenata prima kao privatne podatke, a ostatak kao javne. Javni podaci su ID, organizacija koja je vlasnik *asseta* i opis, a privatni su boja i veličina. Hyperledger Fabric privatne podatke implementira kao tranzijentnu *hashanu* (kriptografski šifriranu) mapu koju će kasnije moći pročitati jedino trenutni vlasnik *asseta*. Ta se tranzijentna mapa pohranjuje u transakciju putem metode setTransient. Bitno je spomenuti podatak „salt“, koji je spremljen zajedno s veličinom i bojom. Pod *salt* se upisuje neki nasumični niz slova i brojeva duljine 40 znamenki, a ostvaruje dodatnu sigurnost privatnih podataka. Naime, s obzirom da su privatni podaci pohranjeni na *blockchain* kao *hash* koji jedino vlasnička organizacija može dešifrirati, u teoriji bi neka druga organizacija mogla *brute force* metodom pogoditi boju i veličinu *asseta* tako da samo pogađa razne kombinacije boja i veličina dok ne pogodi onu koja je jednaka kao *hash* spremljen na *blockchain*. Dodavanjem dugačkog nasumičnog niza znamenki u *hash*, ostvaruje se da je nemoguće dešifrirati pohranjeni *hash* u realnom vremenu.

Nakon toga, dovoljno je pozvati metodu submit kako bi transakciju poslali *blockchain* mreži s identifikacijskim brojem i opisom poslanim kao argumentima.

Čitanje *asseta*

Čitanje *asseta* obavlja se na sličan način samo u obrnutom smjeru. Pritiskom na dugme „Read asset“ u prozoru aplikacije poziva se metoda readAsset, prikazana na Slici 4.

Ova metoda, kao i prošla, prvo dohvaća *contract* a zatim briše postojeće podatke iz grafičkog sučelja. Nakon toga, poziva metodu evaluateTransaction s argumentima „readAsset“ i identifikacijskim brojem *asseta* kojeg želimo pročitati. Ovdje je bitno uočiti razliku u odnosu na stvaranje *asseta*. Naime, operacije čitanja s *blockchaina*, odnosno dohvaćanja nekog podatka nije potrebno zapisati na *blockchain* pa ne koristimo istu

metodu. Metoda evaluateTransaction samo čita neki podatak s *blockchaina,* a na njega ništa ne zapisuje.

```
protected static void readAsset(Gateway gateway, String assetId, JTextArea id, JTextArea owner, JTextArea desc) {
    try {
        Network network = gateway.getNetwork("mychannel");
        Contract contract = network.getContract("secured");

        id.setText("");
        owner.setText("");
        desc.setText("");

        byte[] result;

        result = contract.evaluateTransaction("readAsset", assetId);
        console.append(new String(result) + "\n");

        JSONObject resultAsJSON = new JSONObject(new String(result));
        id.append(resultAsJSON.getString("assetID"));
        owner.append(resultAsJSON.getString("ownerOrg"));
        desc.append(resultAsJSON.getString("publicDescription"));

        console.append("Successfully read public data of asset " + assetId + ". ");
    } catch(Exception e) {
        console.append(e.toString() + "\n");
    }
}
```

Slika 5. Kod metode readAsset

Nakon slanja transakcije slijedi čitanje primljenih podataka i prikazivanje na grafičkom sučelju. Ova metoda čita samo javne podatke o *assetu* i može ju koristiti bilo koja organizacija neovisno je li vlasnik *asseta*. Za privatne podatke koristi se metoda readPrivateAsset, koja radi na istom principu kao i readAsset, osim što poziva transakciju „GetAssetPrivateProperties“.

Prodajna ponuda

Organizacija koja posjeduje neki *asset* može se odlučiti staviti ga na prodaju. To čini pozivanjem metode agreeToSell i upisivanjem željene prodajne cijene u prozor aplikacije. Kod metode agreeToSell prikazan je na Slici 5.

Ova metoda poziva transakciju „AgreeToSell“ iz pametnog ugovora Secured asset transfer. S obzirom da ta transakcija zapisuje prodajnu cijenu *asseta* na *blockchain*, ponovo je potrebno koristiti metodu submitTransaction za slanje na *blockchain*.

```
protected static void agreeToSell(int org, Gateway gateway, String assetId, String sellPrice, JTextArea tradeId) {
    try {
        Network network = gateway.getNetwork("mychannel");
        Contract contract = network.getContract("secured");

        tradeId.setText("");

        Transaction createAsset = contract.createTransaction("agreeToSell");
        String assetId2 = assetId;
        String sellPrice2 = sellPrice;

        JSONObject assetProperties = new JSONObject();
        Field changeMap = assetProperties.getClass().getDeclaredField("map");
        changeMap.setAccessible(true);
        changeMap.set(assetProperties, new LinkedHashMap<>());
        changeMap.setAccessible(false);
        Map<String, byte[]> transientData = new HashMap<String, byte[]>();
        assetProperties.put("asset_id", assetId2);
        assetProperties.put("trade_id", RandomStringUtils.randomAlphanumeric(40));
        assetProperties.put("price", Integer.parseInt(sellPrice2));

        transientData.put("asset_price", assetProperties.toString().getBytes("utf-8"));

        console.append("Setting sell price for asset " + assetId + " to " + sellPrice + "\n");

        createAsset.setTransient(transientData);

        createAsset.submit(assetId);

        tradeId.append(assetProperties.getString("trade_id"));

    } catch(Exception e) {
        console.append(e.toString() + " in function agreeToSell\n");
    }
}
```

Slika 6. Kod metode agreeToSell

Ponuđena prodajna cijena *asseta* ostvarena je na sličan način kao i privatni podaci o *assetu*. Identifikacijska oznaka *asseta* koji je na prodaju, tražena prodajna cijena i ID ponude spremaju se u *hash* koji će biti pohranjen na *blockchain*. Ovdje se ID ponude implementira slično kao *salt*, odnosno kao nasumičan niz slova i brojki koji ponovo osigurava sigurnost tih podataka.

Za kupovinu i prodaju *asseta* u ovom pametnom ugovoru koristi se sljedeći princip. Recimo da Org1 stavi svoj *asset* na prodaju i očekuje za njega 50 nekih zamišljenih *tokena*. Aplikacija će stvoriti neki nasumični ID ponude, te ga zajedno s identifikacijskim brojem *asseta* i cijenom staviti na *blockchain* u obliku *hasha*. Zatim, kada bi Org2 htjela kupiti isti taj *asset*, morala bi na neki način kontaktirati Org1 i saznati njihovu ponuđenu cijenu i ID ponude. Tek tada Org2 može pozivom transakcije „AgreeToBuy“ ponuditi cijenu za koju je spremna kupiti *asset*, i mora priložiti ID ponude koji je dala Org1. Transakcija „AgreeToBuy“ poziva se u metodi agreeToBuy a potrebni ID ponude i cijena se upisuju preko grafičkog sučelja.

Transfer *asseta*

Nakon što su Org1 i Org2 obje zapisale svoje ponude na *blockchain*, moguće je pokušati obaviti transfer *asseta* novom vlasniku. To se obavlja metodom transferAsset prikazanoj na slici 6.

```
protected static void attemptTransfer(int org, Gateway gateway, String assetId, String buyerOrg) {
    try {
        Network network = gateway.getNetwork("mychannel");
        Contract contract = network.getContract("secured");

        JSONObject assetProperties = new JSONObject();
        JSONObject priceAsJSON = new JSONObject();

        Field changeMap = assetProperties.getClass().getDeclaredField("map");
        changeMap.setAccessible(true);
        changeMap.set(assetProperties, new LinkedHashMap<>());
        changeMap.setAccessible(false);

        changeMap = priceAsJSON.getClass().getDeclaredField("map");
        changeMap.setAccessible(true);
        changeMap.set(priceAsJSON, new LinkedHashMap<>());
        changeMap.setAccessible(false);

        JSONObject fetchProperties = new JSONObject(new String(contract.evaluateTransaction("GetAssetPrivateProperties", assetId)));
        JSONObject fetchPrice = new JSONObject(new String(contract.evaluateTransaction("GetAssetSalesPrice", assetId)));

        assetProperties.put("object_type", fetchProperties.getString("object_type"));
        assetProperties.put("asset_id", fetchProperties.getString("asset_id"));
        assetProperties.put("color", fetchProperties.getString("color"));
        assetProperties.put("size", fetchProperties.getInt("size"));
        assetProperties.put("salt", fetchProperties.getString("salt"));

        priceAsJSON.put("asset_id", fetchPrice.getString("asset_id"));
        priceAsJSON.put("trade_id", fetchPrice.getString("trade_id"));
        priceAsJSON.put("price", fetchPrice.getInt("price"));

        console.append("Attempting to transfer " + assetId + " to buyer " + buyerOrg + "MSP at the price of " + priceAsJSON.getInt("price") + "\n");

        Transaction attemptTransfer = contract.createTransaction("TransferAsset");
        attemptTransfer.setEndorsingPeers(peers);

        Map<String, byte[]> transientData = new HashMap<String, byte[]>();

        transientData.put("asset_properties", assetProperties.toString().getBytes("utf-8"));
        transientData.put("asset_price", priceAsJSON.toString().getBytes("utf-8"));

        attemptTransfer.setTransient(transientData);

        attemptTransfer.submit(assetId, buyerOrg + "MSP");
        console.append("Org" + org + " successfully transfered " + assetId + " to buyer " + buyerOrg + " for " + priceAsJSON.getInt("price"));

    } catch(Exception e) {
        console.append(e.toString() + " in function attemptTransfer\n");
    }
}
```

Slika 7. Kod metode attemptTransfer

Ova metoda zahtijeva malo više posla nego prijašnje. Nju može pozvati jedino trenutni vlasnik *asseta*, a njen princip rada je sljedeći. Poziva se transakcija „TransferAsset“ koja kao argumente prima identifikacijsku oznaku *asseta* i ime organizacije na koju se pokušava prebaciti *asset*. Osim toga, transakciju zahtjeva i tranzijentnu mapu koja ovoga puta sadrži i privatne podatke o *assetu* i privatne podatke o prodajnoj ponudi. Ove se dvije mape u pametnom ugovoru uspoređuju s već zapisanim *hashem* na *blockchainu* kako bi se još jednom utvrdilo prodaje li organizacija zaista točno taj *asset* na koji se misli.

Osim toga, uspoređuju se i ponuda koju je prodavač zapisao s onom koju je na *blockchain* zapisao potencijalni kupac. Kako bi transfer bio uspješan, potrebno je da *hash*

prodavačeve ponude bude identičan kupčevoj. To može biti slučaj jedino onda kada su i kupac i prodavač oboje zapisali jednaku cijenu i ID ponude. Kada se prodavač i kupac ne bi složili oko cijene, *hash* ponuda bi bio drukčiji, te bi pokušaj transfera bio odbijen.

Još jedna novost u ovoj metodi je metoda setEndorsingPeers pozvana nad transakcijom. Ovime se postavljaju *peerovi* koji trebaju sudjelovati u procesu potvrđivanja transakcije. U ovom slučaju, to su *peerovi* onih organizacija koje sudjeluju u transferu *asseta*.

Ostale metode

Ostale metode se samo po detaljima razlikuju od već spomenutih. Primjerice, tu je još changeDescription, koji funkcionira slično kao createAsset samo puno jednostavnije jer samo mijenja opis *asseta*. Nadalje, imamo getAssetSalesPrice i getAssetBidPrice koje rade slično kao readAsset, samo što čitaju privatne podatke o ponudi. Slično kao getAssetPrivateProperties, organizacija može pročitati jedino svoju vlastitu ponudu jer je privatno zapisana.