Analiza tehnologije Blockchain

Pejčić, Lovro

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni preddiplomski studij računarstva

ANALIZA TEHNOLOGIJE BLOCKCHAIN

Završni rad

Lovro Pejčić

Osijek, 2018

Sadržaj

1. UVOD

Nastupilo je vrijeme razvitka brojnih grana IT sektora među kojima je i lanac blokova. On je relativno nova tehnologija koja omogućuje siguran prijenos podataka između dviju stranaka bez posredovanja treće te nam pruža preuzimanje kontrole nad našim sredstvima. Ona također potiče globalizaciju, jer omogućuje brže i sigurnije globalne transakcije. Kada se govori o tehnologiji lanca blokova teško je reći govorili se o Bitcoin[1] lancu blokova, Ethereum lancu blokova, virtualnoj valuti općenito, pametnim ugovorima ili o nečemu sličnome. U suštini, zajednička tehnologija koja se koristi kod ovih pojmova je raspodijeljena „glavna knjiga". Odnosno podatci koji se spremaju na različite poslužitelje, a svaki od poslužitelja u komunikaciji ima spremljene iste podatke. Kada se govori o podatcima to mogu biti podatci bilo koje vrste, iako lanac blokova najčešće sadrži financijske podatke i podatke svih transakcija u mreži. On je postao poznat u svijetu zbog popularnosti kriptovaluta, ali uporaba tehnologije lanca blokova nije ograničena samo na njih.

Cilj ovog rada je teorijska analiza tehnologije lanaca blokova (eng. *blockchain*) i pokazivanje da on nije isto što i kriptovaluta, iako je mišljenje većine da su pojmovi bitcoin i lanac blokova istoznačni. Stoga će u zadnjem poglavlju biti prikazano korištenje tehnologije lanca blokova na primjeru poslovne mreže.

U poglavlju 2 obrađena je povijest nastanka lanca blokova, tehnologija lanaca blokova, konsenzus i neke od njegovih metoda, zatim struktura blokova te algoritmi korišteni pri „šifriranju" podataka. U poglavlju 3 objašnjena je primjena tehnologije lanca blokova na primjeru kriptovaluta i pametnih ugovora, te okoline i alati korišteni za razvijanje lanca blokova. Na koncu, u poglavlju 4, prikazana je primjena korištenja lanca blokova na primjeru poslovne mreže.

1.1. Zadatak završnog rada

Zadatak rada je opisati način rada i najznačajnije primjene tehnologije lanca blokova, analizirati glavne izazove u postojećim i budućim primjenama, te prikazati korištene alate i okoline za zasnivanje sustava. Na prikladnom praktičnom primjeru bit će prikazan i analiziran postupak razvoja i primjene rješenja.

[1] Kada se riječ *BITCOIN* piše velikim početnim slovom odnosi se na tehnologiju, a kada je pisano malim početnim slovom misli se na kriptovalutu.

2. TEHNOLOGIJA LANCA BLOKOVA

Prema [1], lanac blokova (eng. *blockchain*) predstavlja podijeljenu strukturu podataka odnosno listu informacija podijeljenih između svih čvorova koji se nalaze u sustavu. S obzirom da baza podataka nije sačuvana na jednom mjestu odnosno poslužitelju, ona je decentralizirana. Upravo je decentralizacija glavna ideja lanca blokova. Umjesto da svatko na svijetu ima svoje knjige i vodi zasebne evidencije transakcija, glavna knjiga je jedna, sadrži sve transakcije, javna je i u vlasništvu svih. Nastala je kombinacijom *peer-to-peer* (P2P) mreže i raspodijeljenog poslužitelja koji obilježava transakcije vremenskim žigom. Lanac blokova možemo zamisliti kao datoteku u kojoj su podaci posloženi i strukturirani. U svaki blok se može zapisati određen broj podataka odnosno transakcija. Kada se blok popuni kreira se novi, a samim time se tvori neprekinuti lanac blokova koji su međusobno povezani. Kada netko pošalje primjerice bitcoin s jedne adrese na drugu adresu, transakcija je vremenski obilježena i zabilježena kod svakog sudionika sustava. Nitko ne može prevariti sustav i poslati nešto što nema u vlasništvu, jer je stanje računa javno, sinkronizirano s ostalim sudionicima sustava i evidencijom transakcija, a pravila su definirana na početku i implementirana kroz programski kod. Na ovaj način, Satoshi Nakamoto je prvi dokazao da problem dvostruke potrošnje (eng. *double spending problem*) može biti riješen bez treće strane, odnosno posrednika kojem obje strane vjeruju. Nakon što se transakcija odobri, posebni čvorovi šalju podatke po cijeloj mreži o transakciji i svaki server transakciju zapisuje kod sebe. Lanac blokova može biti javan i privatan. On može biti javan na dva načina. Prvi način je da svi mogu zapisivati i čitati podatke. Drugi način je da svi mogu čitati podatke, a samo ovjerene osobe zapisivati. Za razliku od javnog, privatni lanac blokova je onaj kojem može pristupiti samo odobrena osoba koja posjeduje određeni *token* s kojim može zapisivati transakcije. To može biti prednost u velikim organizacijama koje će iskoristiti te mogućnosti lanca blokova, a tim podatcima javnost neće moći pristupiti. Kakav god lanac blokova bio, mora sadržavati način za postizanje dogovora odnosno konsenzusa među brojnim sudionicima da bi provjerili valjanost podataka.

2.1. Povijest lanca blokova

Razvitak tehnologije lanca blokova počeo je već 1991. godine. Struktura slična lancu blokova prvi put je spomenuta u istraživačkom radu Stuarta Habera i W. Scotta Stornetta pod imenom *How to Time-Stamp a Digital Document* [2]. Implementirali su sustav u kojemu su datoteke vremenski obilježene kako bi spriječili petljanje s podatcima i mijenjanje istih. Godine 1992. Bayer, Stornetta i Haber su ukomponirali Merkleovo stablo u dizajn kako bi poboljšali

učinkovitost dopuštajući spremanje većeg broja dokumenta u jedan blok. Prvi lanac blokova je koncipirala osoba ili skupina ljudi pod imenom Satoshi Nakomoto 2008. godine. Koncept lanca blokova izdan je u članku *Bitcoin: A Peer to Peer Electronic Cash System* iste godine. Iduće godine, tehnologija lanca blokova implementirana je kao jezgra kriptovalute zvane bitcoin, čime je bitcoin postao prva kriptovaluta koja je riješila problem dvostruke potrošnje. Godine 2014. razvijen je Blockchain 2.0 koji između ostalog omogućuje pisanje sofisticiranijih pametnih ugovora i monetiziranje vlastitih informacija. Kako je navedeno u [3], u kolovozu 2014. godine veličina datoteke Bitcoin lanca blokova, koja bilježi sve transakcije, dosegla je 20 GB, a iste godine svijet je shvatio da on ima i druge primjene osim u kriptovalutama. Veličina te datoteke se u siječnju 2017. godine popela na 100 GB prema izvoru [4].

2.2. Problem bizantskih generala

Kako bi se lakše shvatio način funkcioniranja konsenzusa koji su potrebni za funkcioniranje lanca blokova, možemo si predočiti veliku srednjovjekovnu vojsku koja je opkolila grad s mnogo divizija i generala na čelu tih divizija. Oni moraju postići konsenzus kako bi izveli koordinirani napad na grad. Niti jedan od generala ne vidi ostale generale. Mogu komunicirati samo preko glasnika, što dovodi do problema razmjene informacija. Istovremeno ne postoji sigurnost da su svi generali lojalni vojsci. Neki od generala su izdajnici koji su potplaćeni od strane neprijateljske vojske i ne žele da se konsenzus postigne između lojalnih generala. Lojalni generali žele napasti grad, ali im je potrebna istovremena većina. Dakle, potreban je način da svi odluče o istoj taktici napada, a da manji broj izdajica krivim informacijama ne prisili lojalne generale da uvaže pogrešan plan. Lojalni će postupiti u skladu s algoritmom dok će izdajice postupiti kako god oni žele. Problem bizantskih generala prikazuje izazov postizanja konsenzusa u raspodijeljenim, decentraliziranim sustavima u kojima ne postoji dobar protok informacija i u kojem postoje neprijatelji sustava. Pretpostavka je da su generali sudionici raspodijeljene mreže koja je bazirana na lancu blokova. Glasnici su način komuniciranja unutar mreže. Lojalni generali su sudionici mreže koji žele da mreža funkcionira prihvaćanjem samo točnih informacija. Izdajnici su sudionici koji žele lažirati informacije ili unijeti netočne informacije u bazu podataka, a cilj većine, odnosno lojalnih generala, je da odluče je li informacija koja se unosi ispravna ili ne. Upravo bi ovaj problem mogla riješiti tehnologija lanca blokova na način da bi automatski isključila izdajice iz sustava postizanjem konsenzusa.

2.3. Konsenzus

Tehnologiji lanaca blokova potrebne su konsenzusne metode, jer je decentralizirani *peer-to-peer* sustav koji nema centralni autoritet, za razliku od centraliziranih sustava u kojima glavne

odluke donese centralni autoriteti. Konsenzus u lancu blokova predstavlja proces u kojemu su transakcije i njihove evidencije u okviru cijele mreže i kod svih sudionika sinkronizirane, kako bi se osiguralo da su knjige ažurirane samo onda kada su odobrene od strane sudionika, te da su ažurirane uvijek s istim transakcijama i u istom redoslijedu. Nedostatak povjerenja u lancu blokova sustavima čini okosnicu potrebnu za postizanje konsenzusa u okviru mreže. Zbog toga što jednom uneseni podatci u lanac blokova postaju nepromjenjivi i zato što svaki sudionik mreže može unijeti nove podatke, neophodno je da se svi sudionici mreže slože prije unosa podataka. Postoji nekoliko metoda postizanja konsenzusa, koje će biti objašnjene u sljedećim odlomcima, a to su: *Proof of Work, Proof of Stake, Proof of Authority, Practical Byzantine Fault Tolerance, Delegated Proof of Stake* i *Proof of Importance*.

2.3.1. Proof of Work

Proof of Work je najkorištenija metoda za postizanje konsenzusa. Kako bi odobrio blok, sudionik koji održava mrežu, odnosno čvor (eng. *node*), mora riješiti vrlo kompleksan matematički zadatak. Svrha rješavanja matematičkih zadataka krije se u tome da simulira rad – odnosno da uređaj troši električnu energiju. Za uzvrat, čvor je nagrađen određenom količinom kriptovalute uključujući i cijenu transakcije. Taj proces predstavlja rudarenje (eng. *mining*). Jedini način na koji bi rudari tj. čvorovi koji održavaju mrežu mogli varati sistem je taj da posjeduju 51% ukupne računalne snage cijele mreže. Čak i tada je nemoguće mijenjati već unesene transakcije, moguće je jedino zaustaviti naredne transakcije. Kao što je rečeno u [5], što više računalne snage čvor ima, veća je vjerojatnost da će prvi riješiti matematički problem i zauzvrat dobiti nagradu, upravo to je razlog zašto u sustavima lanca blokova dolazi do velikog udruživanja. Ipak, ova metoda ima svoje mane, jer što je mreža veća potrebno je više energije za potvrđivanje transakcije, više vremena kako bi se transakcija potvrdila što znači manje transakcija u sekundi. Upravo su te mane razlog zbog kojih se ova metoda preispituje.

2.3.2. Proof of Stake

Kod ove metode ne postoji rudarenje, već kao što naziv kaže postoji dokaz o ulogu. Neki čvorovi obrađuju transakcije dok ih drugi potvrđuju. Kako bi se izbjegli ili kaznili pokušaji varanja, čvorovi zaključavaju jedan dio svojih sredstava u virtualni sef s jednostavnim digitalnim potpisom o vlasništvu. Kao što je rečeno u [6], mreža po sistemu lutrije bira sudionika da potvrdi podatke unesene u bazu podataka na osnovu njegovog uloga. U slučaju da čvor pokuša prevariti sustav, ulog mu biva oduzet. Slično rudarenju, što je veći ulog veća je šansa da osoba koja je uložila više sredstava potvrdi blok transakcija i kreira sljedeći, a samim time više gubi ukoliko

pokuša prevariti sustav. Ovakav sustav je često na meti kritike da je zapravo centraliziran, jer što je netko duže na mreži ima veći udio te samim time i veću kontrolu nad mrežom.

2.3.3. Proof of Authority

Dokaz autoriteta (eng. *Proof of Authority*) je metoda postizanja konsenzusa unutar lanca blokova u kojem se odabiru određeni korisnici tj. čvorovi da bi ovjerili blokove. Oni ne ulažu ni svoju struju ni svoj novac, već samo svoj „ugled“. *Proof of Authority* funkcionira na način da se prije stvaranja mreže lanca blokova odaberu čvorovi koji će ovjeravati potvrđene blokove odnosno rudariti. Ti odabrani čvorovi će predati određeni dokaz identiteta koji ih čini pouzdanima. Potrebno je izabrati više takvih čvorova prije nego sustav postane funkcionalan. Ukoliko određeni čvor ne ovjeri blok prema zadanim pravilima biva izbačen iz mreže kao u *Proof of Stake* sustavu. Za razliku od *Proof of Stake* metode u kojoj čvor može ponovno postati član ulaganjem, u *Proof of Authority* sustavu to nije moguće, jer je povjerenje čvora bilo utemeljeno na ugledu kojeg je izgubio. Upravo je to činjenica na kojoj se temelji ova metoda, ukoliko korisnik odnosno čvor radi protiv drugih – više ne može sudjelovati u procesu. Ovakva je metoda postizanja konsenzusa jedino korisna u zatvorenim i privatnim lancima blokova. Zrakoplovne tvrtke, banke i osiguravajuće kuće su idealni kandidati za ovakav sustav lanca blokova. Pojedini se članovi udruženja ne moraju poznavati niti si vjerovati, no zajednička suradnja im najviše koristi, jer na takav način smanjuju konkurenciju. Ukoliko jedan član npr. Croatia Airlines prekrši pravila biva izbačen iz sustava i više ne može sudjelovati, čime gubi mnogo više nego da je zadržao ugled i ostao u sustavu.

2.3.4. Practical Byzantine Fault Tolerance (PBFT)

PBFT je najzastupljenija metoda postizanja konsenzusa u tzv. industrijskim odnosno privatnim sustavima lanca blokova i jedno od mogućih rješenja prethodno navedenog problema bizantinskih generala. Ovom metodom svaki sudionik održava svoje interno stanje. Kada dobije poruku od „glasnika“ on koristi informacije iz poruke usporedno sa svojim internim stanjem i izvršava operaciju. Izvršena operacija omogućuje sudioniku da donese zaključak o primljenoj poruci. Nakon toga on dijeli svoju odluku sa sudionicima sustava te se konsenzus donosi na osnovi svih odluka koje su sudionici poslali. Uporabom ove metode se lakše postiže konsenzus,

ali na štetu anonimnosti. To je razlog zašto ga koriste sustavi lanca blokova bazirani na članstvu. Kao što je rečeno u [7], neki od primjera su Hyperledger[2] i Ripple[3].

2.3.5. Delegated Proof of Stake (dPoS)

U ovoj metodi postoje dvije vrste čvorova: svjedoci i delegati. Svjedoci potvrđuju transakcije i dobivaju isplatu provizije, dok delegati donose odluke o pravilima, provizijama, veličini blokova i drugim karakteristikama. Obje vrste čvorova se biraju na osnovi glasanja većine i pritom se biraju odvojeno, što potencijalno rješava sukob interesa. Kao što je rečeno u [8], prvih sto svjedoka je plaćeno za svoje usluge, a samo prvih dvadeset ima stalnu plaću od strane mreže za svoj rad. Postoje i rezervni svjedoci čime se stvara pritisak na postojeće svjedoke kako bi oni što bolje i što efikasnije obavljali svoj posao. Delegati glasaju i snaga njihovog glasa ovisi o broju *tokena* koje imaju u svom posjedu. To znači da delegati koji imaju više *tokena* više utječu na mrežu od onih koji ih imaju manje.

2.3.6. Proof of Importance

Proof of Importance je inačica *Proof of Stake* metode u kojoj svaki sudionik ulaže fiksni iznos svoje valute kako bi postao čvor. Vjerojatnost da će njega sustav izabrati da kreira novi blok i dobije nagradu ovisi o dokazu o važnosti. Na dokaz o važnosti direktno utječe koliko zapravo netko koristi mrežu koja je u pitanju. Veliki broj transakcija velikih vrijednosti podiže sudioniku, odnosno čvoru, ocjenu važnosti.

2.4. Blokovi

Lanac blokova (eng. blockchain) je, kao što samo ime govori, lanac međusobno povezanih blokova podataka. Kao što je rečeno u [9], blokove unutar lanca blokova možemo zamisliti kao stranice unutar jedne knjige tako da riječi i rečenice predstavljaju informacije koje se moraju pohraniti. Napisane su na različitim stranicama koje su međusobno spojene po njihovom redoslijedu i broju. Ukoliko želimo provjeriti nedostaje li koja stranica samo trebamo provjeriti redoslijed brojeva na stranicama, isto tako je i s blokovima unutar lanca blokova samo što on koristi specifični sustav koji se nazivaju *hashiranje*, koji će biti detaljnije pojašnjen u nastavku.

[2]Hyperledger – Linuxova inicijativa za razvoj industrijskih sustava lanaca blokova s velikim brojem Fortune 500 kompanija koje su članovi.
[3]Ripple – privatni i zatvoreni sustav lanca blokova za financijske transakcije koji trenutno implementiraju mnoge banke.

2.4.1. Struktura bloka

Blok je struktura podataka u kojoj su zapisane digitalne informacije koje se dijele putem lanca blokova. Oni su organizirani linearno, a nove transakcije se konstantno procesuiraju od strane rudara (eng. *minera*) u nove blokove koji se nadodaju na kraj lanca. Prema izvoru [10], u tablici 2.1 prikazana je struktura bloka koja se sastoji od magičnog broja (eng. *Magic no.*) koji služi programerima da identificiraju početak odnosno kraj bloka u lancu. Ona također sadrži podatke o veličini bloka, zapise koji prikazuju sadržaj podataka u tom bloku i njihov brojač. U strukturi bloka se također nalazi i zaglavlje, koje će biti detaljnije objašnjeno u sljedećem odlomku, a sadrži meta – podatke.

Tablica 2.1. Struktura bloka

Naziv	**Opis**	**Veličina**
Magični broj	0xD9B4BEF9	4 bajta
Veličina bloka	Veličina bloka u bajtovima	4 bajta
Zaglavlje bloka	Meta-podaci o bloku	80 bajtova
Brojač zapisa	Koliko zapisa sadrži blok	1-9 bajtova
Zapisi	Zapisi pohranjeni u bloku	Varijabilno

2.4.2. Zaglavlje bloka

Zaglavlje bloka sastoji se od 80 bajtova podataka tipa *String* koji služe kao dodatne informacije o bloku i povezivanju lanca. U tablici 2.2 napravljenoj prema izvoru [11], prikazana je struktura zaglavlja bloka.

1. Inačica predstavlja trenutnu inačica bloka.

2. Hash prethodnog bloka predstavlja rezultat primjene *hashing* algoritama nad zaglavljem prethodnog bloka lanca. *Hash* bloka je jedinstveni identifikator svakog bloka odnosno podatka. Upravo s tim *hashevima* se blokovi povezuju i osiguravaju da podaci unutar bloka ostanu ne promijenjeni.

3. *Korijen Merkle stabla* tj. binarnog *hash* stabla prikazuje podatke svih zapisa u bloku koji su *hashirani* zajedno.

4. *Vremenska oznaka* predstavlja vrijeme dodavanja bloka u lanac.

5. *Težinska oznaka* predstavlja broj koji regulira koliko dugo treba *minerima* da dodaju novi blok transakcija unutar lanca.

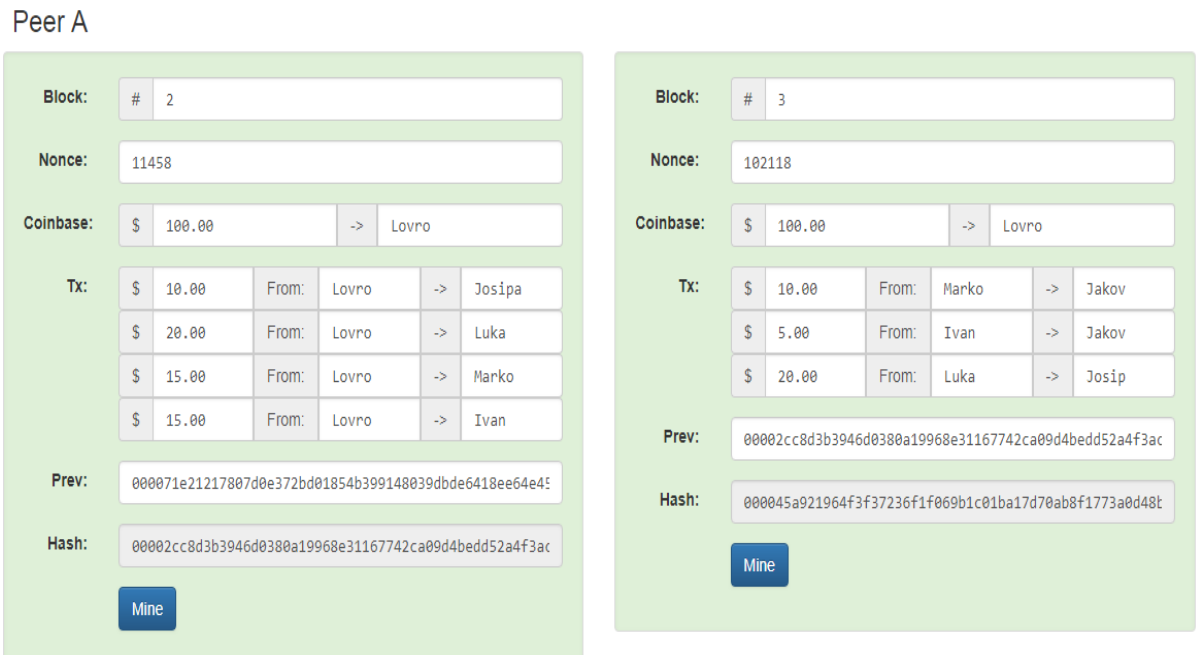
6. *Nonce* broj koji se dodaje na kraj podatka kako bi se izračunao sljedeći *hash*.

Tablica 2.2. Struktura zaglavlja bloka

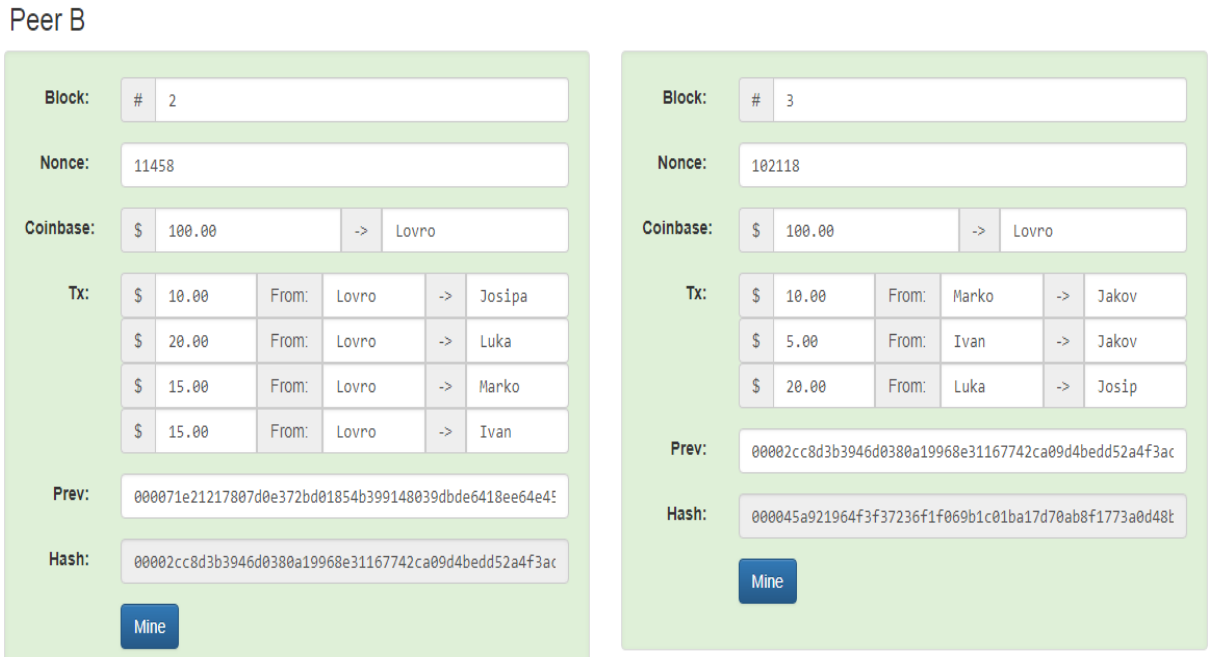
	Naziv	**Opis**	**Veličina**
1.	Inačica	Inačica protokola u vrijeme nastajanja bloka (Specifično za Bitcoin).	4 bajta
2.	*Hash* prethodnog bloka	Referenca na prethodni blok u lancu.	32 bajta
3.	Korijen Merkle stabla	Binarno *hash* stablo koje sadrži informacije o svim zapisima u bloku.	32 bajta
4.	Vremenska oznaka	Vrijeme kada je blok kreiran.	4 bajta
5.	Težinska oznaka	Težina rješavanja algoritma.	4 bajta
6.	*Nonce*	Broj koji pomaže pri rješavanju algoritma.	4 bajta

2.4.3. Povezivanje blokova

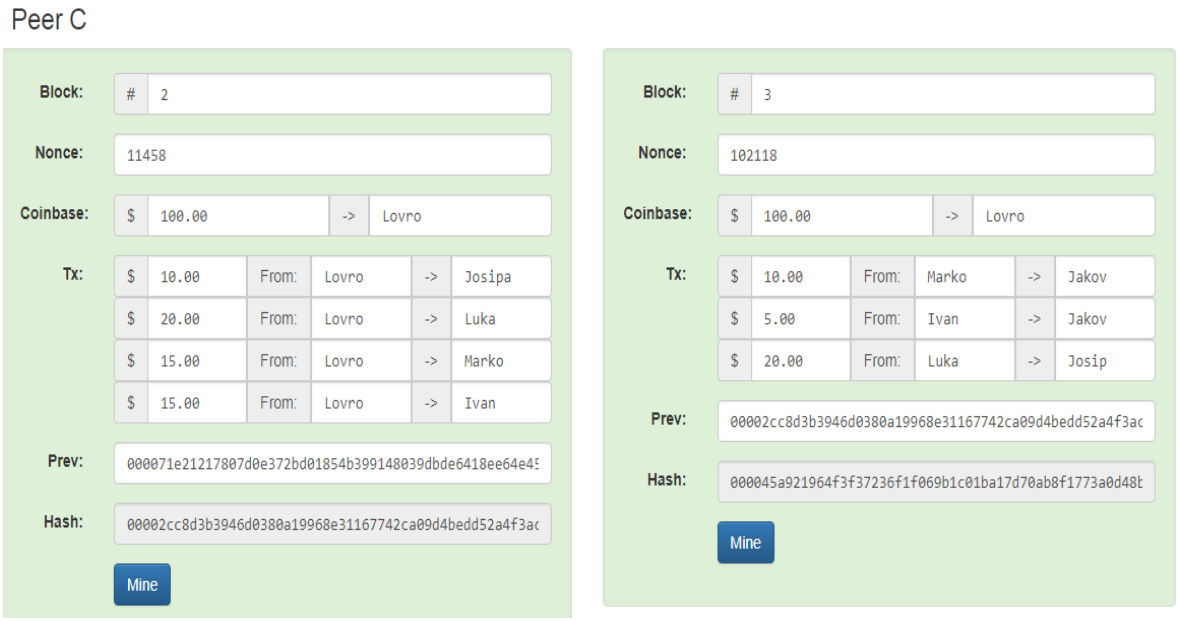
Zapise lanca blokova može se percipirati kao strukturiranu datoteku s mnoštvom podataka koji su logički spremljeni i posloženi. Kao što knjiga ima svoj sadržaj, broj stranica, zaglavlje s naslovom i podnožje s fusnotama tako i lanac blokova ima svoj sadržaj. Npr. podatke o transakciji, *hash* točno određene duljine, meta podatke o bloku te referencu na prijašnji blok. Svaki blok ima određenu količinu podataka koju može primiti. Kada se popuni kreira se novi blok koji se povezuje s prijašnjim i budućim blokom. Kao što se vidi na slici 2.1, *hash* bloka #3 sadrži svoj *hash* i *hash* prethodnog bloka, koji ih povezuje i potvrđuje da nije došlo do izmjene podataka. Istovremeno, na slikama 2.2 i 2.3 tih istih blokova na čvoru B i čvoru C vidljivo je da su blokovi identični. Za kreiranje ovog primjera korišten je program s izvora [12].



Slika 2.1. Prikaz povezanih blokova

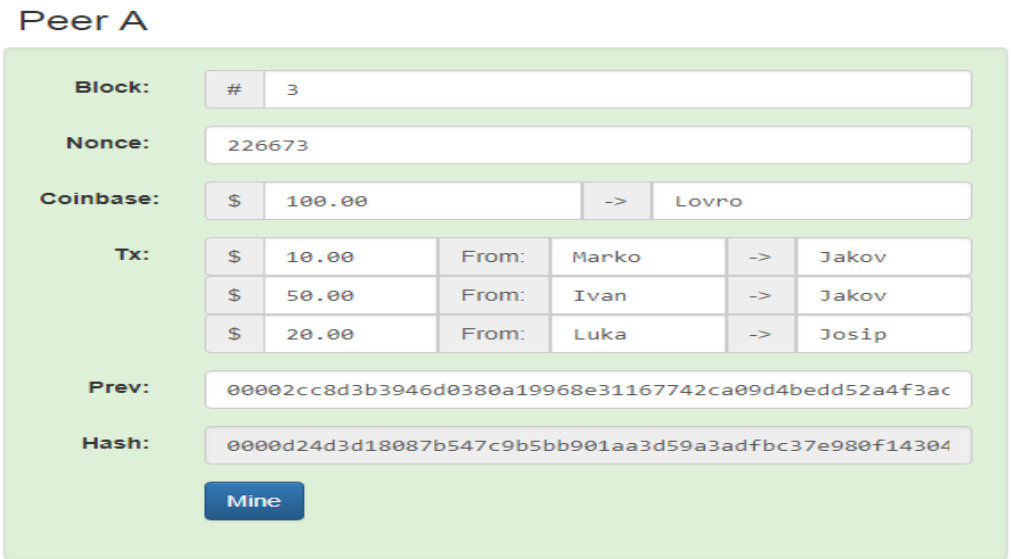


Slika 2.2. Prikaz čvora B



Slika 2.3. Prikaz čvora C

Promjenom podataka unutar bloka, *hash* tog bloka se mijenja, a samim time se izbacuju svi blokovi ispred njega u lancu. Čak i ako se taj blok i svi budući blokovi unutar lanca uspiju „izrudariti“ na vrijeme, *hashevi* tih lanaca se neće podudarati s *hashevima* na drugim čvorovima te će automatski biti odbačeni. Vidljivo je da se *hash* čvora A na slici 2.4 ne podudara s *hashem* bloka #3 na slikama 2.2 i 2.3 koji se nalazi na čvorovima B i C. Zbog toga se vidi da je došlo do izmjene podataka te se taj čvor odbacuje.



Slika 2.4. Prikaz promijenjenog podatka u bloku

2.5. Kriptografija u lancu blokova

Kriptografija je znanost čuvanja podataka na sigurnom. Ona dopušta pohranjivanje informacija i komunikaciju s drugim strankama sprječavajući da neovlaštene stranke razumiju pohranjene informacije ili da razumiju komunikaciju. Šifriranje pretvara razumljiv tekst u nečitljive podatke. Dešifriranjem se nerazumljivi podaci vraćaju u razumljiv tekst. Kao što je rečeno u [13], šifriranje je digitalni ekvivalent zaključavanja lokota, dok je dešifriranje digitalni ekvivalent otključavanja lokota. Postoje dva tipa kriptografije: kriptiranje tajnim ključem i kriptiranje javnim ključem. Kriptiranje tajnim ključem izvodi se pomoću jednog tajnog ključa koji se dijeli između dvije stranke te se za šifriranje i dešifriranje koristi isti ključ. Kod kriptiranja javnim ključem koriste se različiti ključevi za šifriranje i dešifriranje. Stranka ima i javni i privatni ključ. Javni ključ se u pravilu nalazi unutar digitalnog certifikata i dostupan je svima, dok privatni ključ čuva vlasnik. Objekt, kao što je poruka, koji je šifriran nečijim javnim ključem može se dešifrirati samo pridruženim tajnim ključem. U iduća tri odlomka objašnjene su tri najvažnije metode kriptografije lanca blokova.

2.5.1. Digitalni potpis

Digitalni potpis je, poput stvarnog potpisa, postupak kojim se potvrđuje autentičnost i integritet podataka, programske podrške ili digitalnih dokumenata, ali se koriste kriptografske ili matematičke tehnike koje su puno sigurnije od rukom pisanih potpisa. Digitalni potpis je način dokazivanja da poruka potječe od određene osobe, a ne od nekog drugog, primjerice hakera. Što znači, pri posjeti stranica poput Facebooka, preglednik može provjeriti digitalni potpis koji je došao s web stranicom kako bi potvrdio da je doista potekao s Facebooka. U asimetričnim sustavima enkripcije, korisnici generiraju jedan par ključeva, odnosno javni i privatni ključ koji su već objašnjeni. Potpis je uključen u poruku tako da primatelj može provjeriti pomoću javnog ključa pošiljatelja. Također, svaka transakcija koja se izvršava unutar lanca blokova se digitalno potpisuje od strane vlasnika njegovim privatnim ključem. Na taj način osigurava da jedino vlasnik može upravljati podacima, primjerice premjestiti novac s računa.

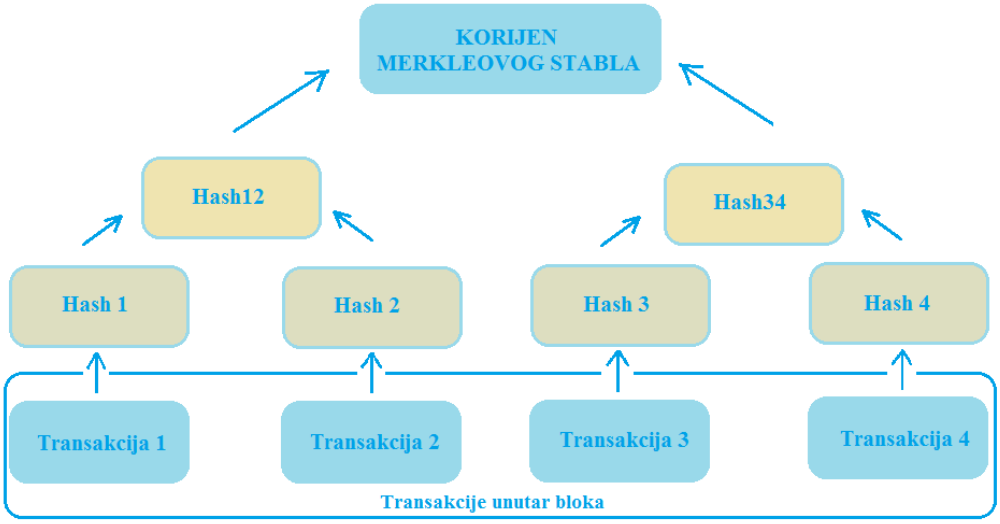
2.5.2. *Hashiranje*

Druga metoda kriptografije je *hashing* funkcija koja uzima ulazne podatke proizvoljne veličine te primjenom nekih od algoritama generira izlaz fiksne duljine koji se naziva *hash*. Bit *hasha* je u tome da uzima ulazne bitove beskonačne duljine, primjenjuje neke izračune te daje izlaze fiksne duljine, npr. 256 bitova. Ulazni podatak može biti bilo koje veličine, uključujući jedan znak, MP3 datoteku, cijelu knjigu, pa čak i cijeli Internet. Postoji nekoliko vrsta *hashing*

algoritama koje se odabiru ovisno o potrebama, jedan od primjera je SHA-2 (*Secure Hashing Algorithm version 2*). *Hashing* se koristi u lancu blokova za prikaz trenutnog stanja. Ulaz je cjelokupno stanje lanca blokova, a dobiveni izlazni *hash* predstavlja trenutačno stanje lanca. Prvi *hash* se izračunava za prvi blok (eng. *Genesis block*) pomoću transakcija unutar tog bloka. Za svaki novi blok koji se generira, koriste se *hashevi* prethodnog bloka i transakcije toga bloka kao ulaz. Na taj način se formira lanac blokova, svaki *hash* novog bloka upućuje na blok prije njega. Takav sustav osigurava da se nijedna transakcija u povijesti ne može promijeniti, jer da se to i dogodi, mijenja se *hash* tog bloka kojem pripada, kao i *hashevi* svih idućih blokova. *Hashing* pruža način da se svi sudionici lanca blokova slože oko trenutnog stanja lanca.

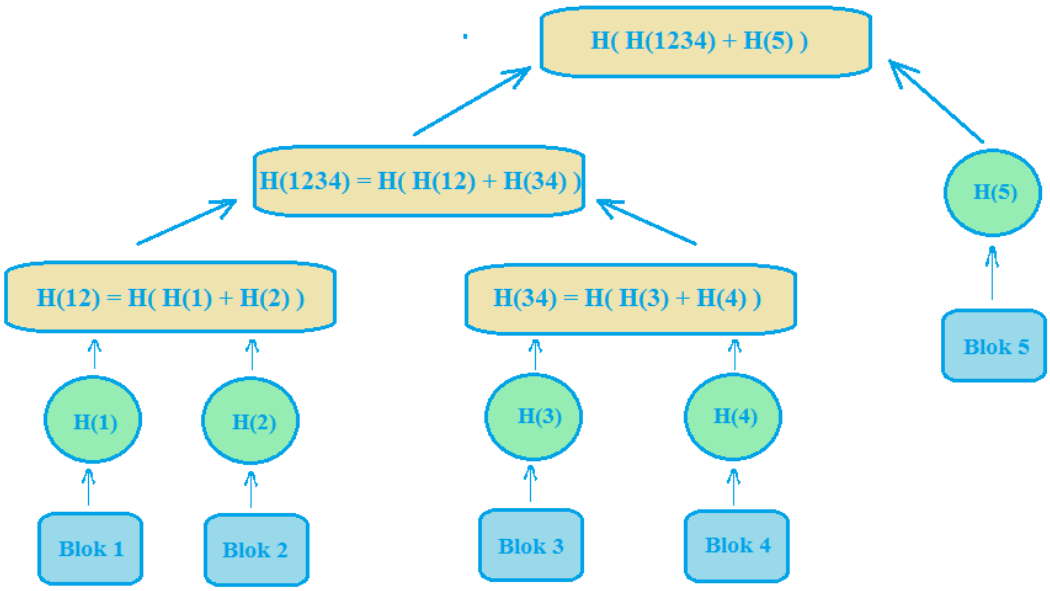
2.5.3. Merkleovo stablo

Prema [14], Merkleovo stablo (binarno hash stablo) je kriptografska metoda čija se struktura koristi za odobravanje velikog broja podataka na učinkovit način. Njegova svrha nije samo provjera nepromijenjenosti podataka primljenih od drugih čvorova mreže, nego i provjera ispravnosti blokova koji se šalju u lanac blokova. Prvi korak u kreiranju Merkleovog stabla je dobivanje *hasha* transakcije kroz primjenu neke kriptografske *hash* funkcije. Nakon čega se *hashevi* svih transakcija sparuju jedan s drugim stvarajući listove stabla. Ovaj se proces ponavlja sve dok ne ostane samo jedan *hash* odnosno korijen Merkleovog stabla. Sparivanjem *hasheva,* Merkleovo stablo osigurava nepromijenjenost transakcija, jer ukoliko se promjeni podatak mijenja se i *hash*, a ako se bilo koji *hash* promjeni to je vidljivo u korijenu Merkleovog stabla. Iz tog razloga korijen Merkleovog stabla predstavlja potpis svih transakcija uključenih u blok, a nalazi se unutar zaglavlja bloka, kao što je rečeno u prethodnom poglavlju. Na slici 2.5 napravljenoj prema izvoru [15], prikazani su podaci unutar jednog bloka pomoću četiri transakcije. Svim transakcijama su izračunati *hashevi* korištenjem neke *hash* funkcije čime oni tvore listove stabla. Iduća razina stabla se dobiva konkateniranjem (spajanjem) parova listova s lijeva nadesno. Zatim slijedi konkateniranje zadnja dva lista Hash12 i Hash34, kad se ono izvrši dobiveni *hash* predstavlja korijen Merkleovog stabla. On predstavlja zapise svih transakcija unutar bloka.



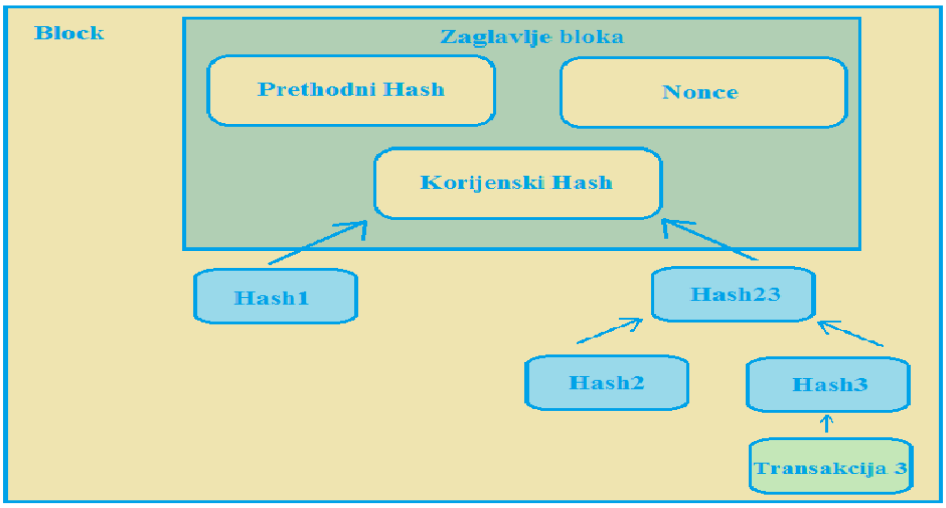
Slika 2.5. Primjer Merkleovog stabla

Svaka sljedeća razina stabla sadrži dvostruko manje čvorova od prethodne. Stablo koje ima 2^n čvorova, odnosno paran broj čvorova, rješava se kao što je prikazano u prethodnom primjeru. Ukoliko se dogodi da je broj čvorova različit od 2^n, rješava se konkateniranjem neparnog čvora samog sa sobom, što je prikazano na primjeru sa slike 2.6. napravljene prema izvoru [16]. Taj slučaj je riješen na sljedeći način: svaki podatak je *hashiran*, a zatim su ti *hashevi* sparuju, osim *hasha* čvora 5 koji je sparen sam sa sobom. Tako je dobiven parni broj listova u sljedećoj razini te se stablo nastavlja rješavati kao ono iz prethodnog primjera.



Slika 2.6. Primjer Merkleovog stabla s neparnim brojem listova

Grane unutar Merkleovog stabla se mogu „rezati“ odnosno brisati kako bi se uštedjelo na memoriji, ali na takav način da je od preostalih grana moguće rekreirati ispravan Merkleov korijen. Postoje dva preduvjeta kada je moguće „rezati“ grane. Prvi uvjet je da su svi izlazi lista odnosno transakcije iskorišteni, dok drugi uvjet govori da se čvor može orezati tek kada su njegova djeca orezana. Na slici 2.7 napravljenoj prema izvoru [17], vidljiva je rekonstrukcija Merkleovog korijena u orezanom Merkleovom stablu.



Slika 2.7 Primjer orezivanja Merkleovog stabla

3. PRIMJENA BLOCKCHAIN TEHNOLOGIJE

3.1. Kriptovalute

Kriptovalute su poseban oblik digitalne, odnosno virtualne valute koje nisu geografski određene za pojedinu državu ili kontinent, a temeljene su na tehnologiji lanca blokova. Nove jedinice kriptovalute se kreiraju rudarenjem kao nagrada korisniku za kreiranje novog bloka. Postoji veliki broj različitih kriptovaluta, tijekom pisanja ovog završnog rada postojalo je 1910 kriptovaluta prema izvoru [18]. Unatoč tome, sve te kriptovalute povezuje šest stavki koje su preduvjet da one budu kriptovalute. Prvi uvjet je da su digitalne, odnosno da postoje samo u računalima, a ne i u stvarnom svijetu. Drugi uvjet je da koriste P2P (*peer-to-peer*) mrežu u kojoj se digitalni novac šalje od jedne do druge osobe *online*. Treći uvjet je da budu globalne, odnosno da se mogu slobodno koristiti diljem svijeta za razliku od stvarnog novca (npr. Hrvatska kuna se može koristiti samo u Republici Hrvatskoj). Četvrti uvjet je da su kriptirane, odakle i dolazi naziv kriptovaluta (kripto-valuta). Što znači da su korisnik kriptovlute i svrha u koju je korištena – tajni. Peti uvjet je da budu decentralizirane što znači da nisu spremljene kod neke treće stranke primjerice banke, već svatko upravlja svojim novcem. Šesti uvjet je da ne treba postojati povjerenje između dviju stranaka koje su uključene u razmjenu zbog načina na koji ovaj sustav funkcionira.

3.1.1. Bitcoin

Bitcoin je prema [19] prva kriptovaluta u povijesti, nastala 2008. godine od osobe ili skupine ljudi koji se nazivaju Satoshi Nakomoto. Predstavljena je člankom pod imenom „*Bitcoin: A Peer-to-Peer Electronic Cash System*“. Zaliha bitcoina je decentralizirana te se može povećavati samo pomoću procesa rudarenja (eng. *mining*). Bitcoin je objavljen kao slobodna programska podrška (eng. *open source software*) što znači da ga svaka osoba može koristiti i modificirati. Zbog toga je došlo do razvoja brojnih kriptovaluta temeljenih na Bitcoinu kao na primjer: Litecoin, Primecoin, Bytecoin i brojnih drugih. Bitcoin koristi *Proof-of-Work* metodu za postizanje konsenzusa koju smo objasnili na početku ovog rada. Također je definiran i maksimalan broj bitcoina koji se može proizvesti, a to je 21 milijun. Prema izvoru [20], do sada je izrudareno preko 17 milijuna bitcoina, a trenutna vrijednost 1 BTC-a je 7373.96 USD.

3.1.2. Ethereum

Ethereum [21] je otvorena platforma lanca blokova koja omogućuje bilo kome da na njoj izgradi i koristi decentralizirane aplikacije kao što su pametni ugovori. Na sam Ethereum lanac

blokova je ugrađena posebna kriptovaluta Ether. Ether nema limit kao bitcoin, ali ima fiksni omjer proizvodnje. Trenutno se izrudari (proizvede) prosječno 5 Ethera svake sekunde. Nakon što Etherum promijeni način rudarenja odnosno postizanja konsenzusa s *Proof-of-Work* na *Proof-of-Stake* proizvodnja će se drastično smanjiti – blizu nule. Upravo time vrijednost Ethera neće disproporcionalno rasti u nedogled kao kod bitcoina, već se postiže stabilnost na tržištu, što je bitno za njegove praktične uporabe. Ether ima dvostruku ulogu: koristi se za plaćanje pri postavljanju ugovora na mrežu i kao sredstvo za nagrađivanja korisnika pri kreiranju bloka. Ethereum skripte odnosno pametni ugovori se pišu u jeziku zvanom Solidity koji dozvoljava pisanje modernih programa. Kada napišemo Solidity program moramo ga poslati u Ethereum lanac blokova kako bi se izvršio. Svaki podatak koji se šalje u lanac blokova treba se platiti određenom količinom Ethera, stoga ljudima nije u interesu pisati velike i neefikasne programe.

3.1.3. VeChain

VeChain [22] je platforma lanca blokova temeljena na Ethereum mreži koja je specijalizirana za praćenje i pohranu informacija o različitim proizvodima. Za svaki proizvod se kreira njegov VeChain identitet na temelju kojeg se može pristupati informacijama o proizvodu. Većinom se koriste QR kodovi i RFID čitači. Označavanjem proizvoda s nekom od navedenih oznaka, proizvođač te potrošači mogu pratiti kompletan put proizvoda kroz proizvodni ciklus i kroz put dostave. Ovo je izuzetno važno za proizvode čiju kvalitetu označuju porijeklo i starost. Na primjer praćenje informacija o prijevozu određene ribe koja se mora skladištiti na strogo određenim temperaturama. Sposobnost VeChaina da se umetne u borbu protiv krivotvorene robe može biti važan dodatak u mnogim industrijama, primjerice tekstilne industrije, prehrambene industrije, automobilske industrije i brojnih drugih. S obzirom da se ovakva platforma većinom koristi u privatnim poslovnim mrežama VeChain primjenjuje drugačiji mehanizam postizanja konsenzusa u mreži koristeći *Proof-of-Authority*. Za razliku od većine drugih sustava lanaca blokova koji koriste *Proof-of-Work* ili *Proof-of-Stake* metode. Također, VeChain je prva platforma lanca blokova koja je sklopila ugovor s određenom vladom. Sklopljen je ugovor s Kineskom vladom te brojnim drugim vodećim svjetskim kompanijama kao što su PricewaterhouseCoopers, DNV GL, Renault grupacijom, KUEHNE+NAGEL, D.I.G., China Unicom te brojim drugima.

3.2. Pametni ugovori

Pametni ugovori su prema [23] jedna od mogućih i najperspektivnijih primjena tehnologije lanca blokova. Naziv „ugovori“ dolazi zbog sličnosti s klasičnim ugovorima u

kojima imamo dvije strane koje prihvaćaju prethodno dogovorene uvjete, ako se oni ispoštuju ugovor je na snazi. U suprotnom ugovor se raskida, a strana koja je prekršila uvjete snosi posljedice. Pametni ugovori sigurnija su inačica ugovora, jer su kreirani na lancu blokova. Oni automatski eliminiraju posrednike između stranaka, poput odvjetnika, banki i slično, a pružaju veću sigurnost. Razlog tome je svojstvo lanca blokova da osigurava nepromjenjivost podataka i da ne ovisi o povjerenju između dviju stranaka koje komuniciraju. Jedini slučaj u kojemu pametnim ugovorima može biti potrebna treća strana je taj da ponekad trebaju informacije iz „stvarnog svijeta". Što nas dovodi do potrebe za sustavima *oracles* koji služe za zapisivanje informacija iz „stvarnog svijeta" u lanac blokova kako bi im pametni ugovori mogli pristupati. Pametni ugovori su programi koji su napisani da automatski kontroliraju prijenos sredstava između dviju ili više strana, nakon što su zadovoljeni prethodno definirani uvjeti. Oni koriste *trigger event*[4] koji ih pokreće, primjerice osoba želi kupiti nekretninu, ali je cijena previsoka. Tada ona u ugovoru definira sebi prihvatljivu cijenu. Ugovor će se automatski pokrenuti kada se dogodi pad cijene nekretnine na željeni iznos. Putem lanca blokova i pametnih ugovora može se izvršiti potpuno automatizirana interakcija iza koje ne stoje nikakvi entiteti. Dakle, bez potrebe za povjeravanjem trećoj stranci, nalazimo potencijalne primjene u državnim, financijskim i telekomunikacijskim sektorima koje mogu dovesti do velikog povećanja efikasnosti. Upravo zbog toga i činjenice da su ugovori temelj organizacije i funkcioniranja društva, pametni ugovori pružaju najperspektivnije mogućnosti primjene tehnologije lanca blokova. Iako je tehnologija lanca blokova najzastupljenija u kriptovalutama, njene najveće mogućnosti leže upravo u pametnim ugovorima.

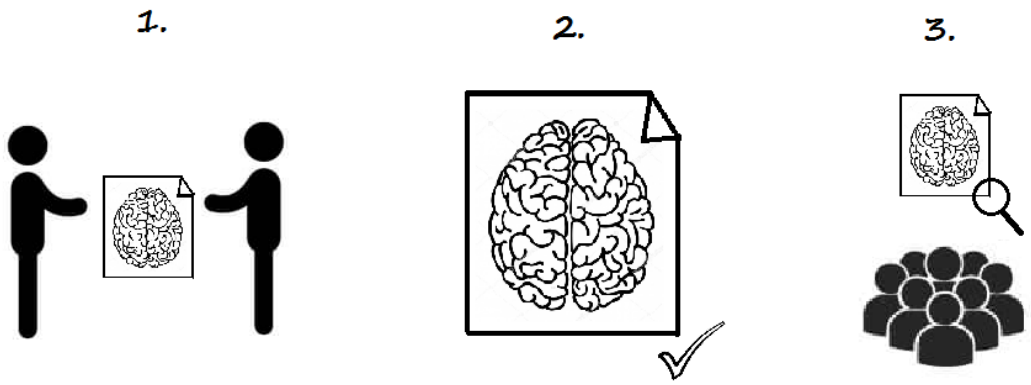
Primjeri pametnih ugovora za automatizaciju ili izbacivanje posrednika su:

- mirovinsko osiguranje: ukoliko državno tijelo (ili liječnik) potpiše da je osoba ispunila uvjete za odlazak u mirovinu, automatski počinje primati mirovinu
- životno osiguranje: ukoliko mrtvozornik unese podatak o preminuloj osobi u lanac blokova, nasljedniku se automatski isplaćuje iznos životnog osiguranja
- automobilsko osiguranje: ukoliko osiguravatelj upiše u lanac blokova da je nastala šteta na vozilu, osigurana osoba automatski dobiva isplatu naknade za nastalu štetu

Postoje tri koraka kreiranja pametnog ugovora, prema izvoru [24]. Prvi korak je zapisivanje pravila razmjene dobara između dviju stranaka unutar programskog koda koji se pohranjuje u

[4] *Trigger event* – događaj koji potiče pokretanje procesa.

lanac blokova. Sadržaj ugovora dostupan je svim korisnicima sustava, dok stranke ostaju anonimne. Drugi korak nastupa nakon *trigger eventa* koji potiče izvršavanje pametnog ugovora prema dogovorenim pravilima. Treći korak dopušta ostalim korisnicima sustava da pretraže lanac blokova kako bi provjerili aktivnosti i rezultate definiranim ugovorom. Na slici 3.1 prikazani su koraci kreiranja pametnog ugovora.



Slika 3.1. Prikaz kreiranja pametnog ugovora

4. OKOLINE I ALATI U LANCU BLOKOVA S PRIMJERIMA KORIŠTENJA

Pojava tehnologija lanca blokova 2009. godine izazvala je interes diljem svijeta. Nakon podrobnijeg istraživanja, zainteresirane stranke su uvidjele da su potrebna velika ulaganja u nepoznatu tehnologiju te im je interes za razvoj vlastitih lanaca blokova splasnuo. Prema istraživanju izvora [25], do sada je samo 1% tržišta uložilo u razvoj vlastitih lanaca blokova, dok 77% trenutno nema interesa za ovu tehnologiju. Upravo zato stvorene su okoline i alati za lakši i jeftiniji pristup te korištenje tehnologije lanca blokova. U nastavku će biti objašnjene neke od mnogobrojnih okolina i alata za razvoj lanca blokova.

4.1. BaaS: Blockchain as a Service

Tvrtke poput IBM-a, Microsofta i Oraclea razvile su vlastite BaaS usluge opisane u [26] kako bi privukle poduzetnike na jednostavnije i jeftinije korištenje tehnologije lanca blokova. I tako je nastao jedan od najznačajnijih alata za primjenu tehnologije lanca blokova – lanac blokova kao usluga ili takozvani BaaS. On sadrži alate potrebne programeru da kreira decentralizirane aplikacije (eng. *dapps*) u sigurnijoj okolini koja podržava više lanaca. Njegov osnovni cilj je omogućavanje programerima da koriste mogućnosti koje im on pruža. Oni tako mogu pokrenuti lanac blokova u samo nekoliko klikova, bez da moraju praviti vlastiti. Upravo je ovaj alat iskorišten za izradu praktičnog primjera u petom poglavlju.

4.2. Coinbase's API

Coinbase's API je također jedan od alata koji pruža mogućnosti razvitka novih aplikacija za kriptovalute i integriranje kriptovaluta u već postojeće aplikacije. Ovaj alat omogućuje širok raspon mogućnosti, od prikupljanja *read-only*[5] podataka pa do kreiranja novih aplikacija. Također, omogućava kreiranje kripto novčanika, kupovine i prodaje kriptovaluta diljem svijeta te pruža niz biblioteka i mobilnih SDK (*Software Development Kit*) biblioteka koje mogu koristiti programerima.

4.3. Embark

Za razliku od Coinbase API-ja u kojem se mogu kreirati razne kriptovalute, Embark je prema [27] stvoren kao razvojna okolina za kreiranje isključivo Ethereum decentraliziranih aplikacija (eng. *daaps*). Što znači da omogućuje kreiranje i implementiranje takvih aplikacija ili

[5] *Read-only* – podaci koji se ne mogu mijenjati, a mogu čitati.

html5 aplikaciju bez poslužitelja koja koristi decentraliziranu tehnologiju. On omogućuje kreiranje novih pametnih ugovora i njihovo korištenje u JS (*JavaScript*) kodu. Također, promatra promjene, pa ako korisnik ažurira ugovor, Embark automatski ažurira ugovore i povezane aplikacije.

4.4. Solc

Prema [28], Solc je kompajler Solidity programskog jezika stvorenog za pisanje pametnih ugovora Ethereum lanca blokova. Veliki broj Ethereum čvorova nativno („urođeno“) sadrži Solc implementaciju, ali je također i zapakiran kao samostalni modul za *offline* kompajliranje. Dakle, Solc pruža slobodu korisnicima hoće li koristiti *web3.eth.compile.solidity* za kompajliranje Solidity datoteka rabeći vlastiti čvor ili koristiti *solc.compile* koji se ne oslanja na bilo kakav vanjski čvor.

4.5. Tierion

Tierion [29] je platforma lanca blokova koja omogućava stvaranje vjerodostojne baze podataka o svim podacima ili procesima koji se nalaze na Bitcoin lancu blokova tako što pruža razvojne alate i API-je za dodavanje podataka glavnoj knjizi (eng. *ledger*). Tierion je također razvio otvoreni standard ChainPoint za pohranu podataka i generiranje računa koji sadrže sve potrebne informacije za potvrdu vjerodostojnosti podataka bez oslanjanja na posrednika.

5. PRIMJER KORIŠTENJA TEHNOLOGIJE LANCA BLOKOVA

Razvitak tehnologije lanca blokova omogućava stvaranje decentraliziranih aplikacija koje mogu olakšati svakodnevni život. Kao što je već rečeno, lanac blokova ne koristi se samo u svrhu kreiranja i korištenja kriptovaluta, već se može rabiti i u druge svrhe. Stoga je u ovom dijelu rada prikazana primjena te tehnologije na primjeru poslovne mreže koja se sastoji od proizvođača, prijevoznika i uvoznika. Ovu poslovnu mrežu definira ugovor između proizvođača i uvoznika koji govori da je uvoznik dužan platiti proizvođaču određenu sumu x za određen broj proizvoda y. Za pošiljke koje su čuvane na preniskim odnosno previsokim temperaturama nadodaju se kazneni bodovi, a pošiljke koje kasne su besplatne. Za kreiranje ovog primjera korišten je *open source* tekstualni editor Visual Studio Code s *plug-inom* za Hyperledger koji omogućuje kreiranje podataka tipa *asset*, *participant* i *transaction* nakon čega je program implementiran u Hyperledger lanac blokova.

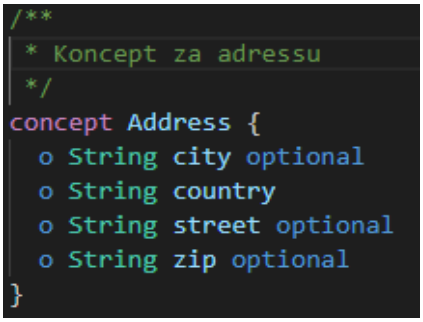
5.1. Postupak razvoja poslovne mreže

Ova poslovna mreža definira korisnike („Grower, Importer, Shipper“), imovinu („Contract, Shipment“) i transakcije („TemperatureReading, ShipmentReceived, SetupDemo“). Na slici 5.1 je vidljiva implementacija korisnika mreže (*participant)* koji sadrži određene informacije kao što su e-mail adresa po kojoj se korisnik identificira, stanje računa i adresu tvrtke. Objekt *Business* apstraktan, što znači da se on ne može kreirati već služi kao predložak za kreiranje korisnika kao što su proizvođač, prijevoznik i uvoznik.

```
/**
 * Tip korisnika u ovoj mreži
 */
abstract participant Business identified by email {
  o String email
  o Address address
  o Double accountBalance
}

/**
 * Proizvođač je tip korisnika u mreži
 */
participant Grower extends Business {
}

/**
 * Prijevoznik je tip korisnika u mreži
 */
participant Shipper extends Business {
}

/**
 * Uvoznik je tip korisnika u mreži
 */
participant Importer extends Business {
}
```

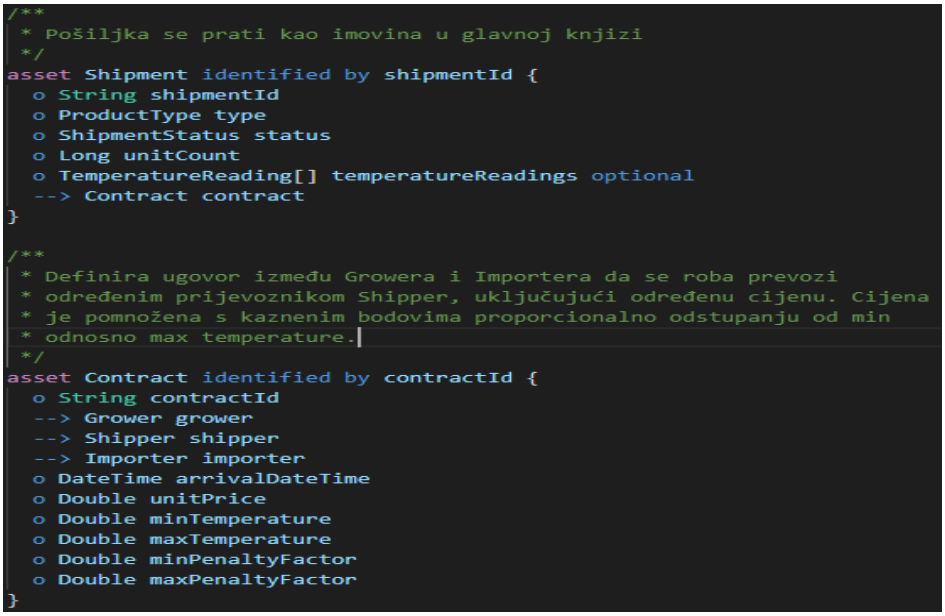
Slika 5.1. Definiranje korisnika mreže

Na slici 5.2 se vidi definiranje adrese koju koriste proizvođač, prijevoznik i uvoznik. Država mora biti upisana od strane korisnika dok su grad, ulica i poštanski broj dodatne opcije. Korišten je tip podatka *concept* koji predstavlja apstraktne klase koje koriste objekti tipa *asset, participant* i *transaction.*



Slika 5.2. Koncept za adresu

Nakon implementacije korisnika slijedi implementacija imovine (*asset*), odnosno pošiljki i ugovora. Na slici 5.3 u gornjem dijelu se vidi implementacija pošiljke s karakteristikama kao što su ID pošiljke, vrsta proizvoda, status pošiljke, broj proizvoda te mogućnost očitavanja temperature ukoliko se radi o lako kvarljivim proizvodima kao što su voće i povrće. Unutar objekta *Shipment* se kreira ugovor između proizvođača, prijevoznika i uvoznika čiju implementaciju prikazuje donji dio slike 5.3. Objekt *Contract* sadrži ID ugovora, vrijeme dolaska robe, cijenu jednog proizvoda, minimalnu i maksimalnu temperaturu te minimalne i maksimalne kaznene bodove koji se nadodaju na cijenu ukoliko dođe do prekoračenja zadane temperature.



Slika 5.3. Definiranje imovine

Sljedeći korak je implementacija transakcija koje se mogu odvijati između korisnika mreže. Na slici 5.4 se vidi kreiranje apstraktnog objekta *ShipmentTransaction* koji služi kao predložak pri kreiranju transakcija za očitavanje temperature i promjeni statusa pošiljke. Unutar *ShipmentTransaction* je vidljivo kreiranje objekta *Shipment* koje te transakcije nasljeđuju kako bi mogle pristupati podacima unutar pošiljke. Na slici 5.5 vidi se transakcija *SetupDemo* koja služi za podizanje mreže, a čija je implementacija napisana unutar funkcije.

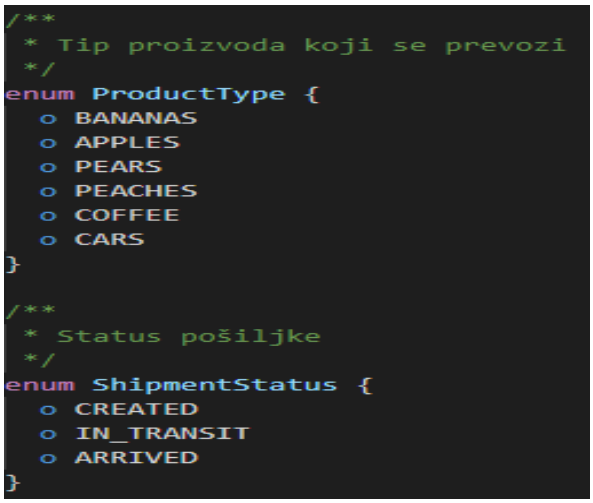
```
/**
 * Transakcija koje je povezana sa Shhipment
 */
abstract transaction ShipmentTransaction {
  --> Shipment shipment
}

/**
 * Temperaturno očitavanje pošiljke od strane nekakvog
  * uređaja unutar kamiona.
 */
transaction TemperatureReading extends ShipmentTransaction {
  o Double centigrade
}

/**
 * Notifikacija koja govori da je pošiljka primljena od strane
 * uvoznika i da novac treba prebaciti na račun proizvođača.
 */
transaction ShipmentReceived extends ShipmentTransaction {
}
```

5.4. Definiranje transakcija

```
/**
 * Služi samo za inicijalizaciju programa.
 */
transaction SetupDemo {
}
```

Slika 5.5. Definiranje transakcije za podizanje sustava

Također, definirani su tipovi proizvoda koji se mogu koristiti te stanja u kojima pošiljka može biti, što se vidi na slici 5.6. Korišten je tip podatka *enum* koji sprema različite konstante tipa *String*.



Slika 5.6. Deklaracija proizvoda i statusa pošiljke

Na slikama 5.7 i 5.8 prikazana je asinkrona funkcija koja se pokreće kada uvoznik primi robu od strane prijevoznika. Korištenjem asinkronih funkcija osigurava se da korisnik sustava ne mora čekati da se određena akcija izvrši do kraja, nego može istovremeno izvršavati više aktivnosti. Funkcija *payOut()* izračunava koliki se iznos s računa uvoznika prebacuje na račun proizvođača. Prvo funkcija provjerava vrijeme u koje je pošiljka stigla i uspoređuje ga s vremenom koje je definirano u ugovoru. Ukoliko pošiljka kasni uvoznik nije dužan platiti odnosno cijena iznosi nula. Nakon toga, program uzima podatke o temperaturi i sortira ih od najniže prema najvišoj temperaturi. Zatim uspoređuje najnižu temperaturu koja je očitana s najnižom temperaturom koja je zadan u ugovoru, isto to radi i s najvišom temperaturom. Ukoliko je došlo do odstupanja temperature izvan granica zadanih u ugovoru, kazneni bodovi se nadodaju na ukupan iznos koji uvoznik mora platiti. Ukoliko su kazneni bodovi toliko veliki, da cijena koju uvoznik mora platiti ode u minus, cijena se postavlja na nulu. Nakon što se svi kazneni bodovi nadodaju na ukupan iznos, on se „skida“ s uvoznikovog računa i uplaćuje na proizvođačev. Na kraju funkcija mijenja status pošiljke u *arrived* čime potvrđuje da je pošiljka stigla.

```
/**
 * Pošiljka je primljena od strane uvoznika
 * @param {org.acme.shipping.perishable.ShipmentReceived} shipmentReceived - ShipmentReceived transakcija
 * @transaction
 */
async function payOut(shipmentReceived) {

    const contract = shipmentReceived.shipment.contract;
    const shipment = shipmentReceived.shipment;
    let payOut = contract.unitPrice * shipment.unitCount;

    console.log('Received at: ' + shipmentReceived.timestamp);
    console.log('Contract arrivalDateTime: ' + contract.arrivalDateTime);

    // postavljanje stanja pošiljke
    shipment.status = 'ARRIVED';

    // ukoliko pošiljka kasni, ne treba platiti
    if (shipmentReceived.timestamp > contract.arrivalDateTime) {
        payOut = 0;
        console.log('Late shipment');
    } else {
        // tražimo min temperaturu
        if (shipment.temperatureReadings) {
            // sortiramo temperature po stupnjevima
            shipment.temperatureReadings.sort(function (a, b) {
                return (a.centigrade - b.centigrade);
            });
            const lowestReading = shipment.temperatureReadings[0];
            const highestReading = shipment.temperatureReadings[shipment.temperatureReadings.length - 1];
            let penalty = 0;
            console.log('Lowest temp reading: ' + lowestReading.centigrade);
            console.log('Highest temp reading: ' + highestReading.centigrade);

            // provjeravamo je li min temperatura krši ugovor
            if (lowestReading.centigrade < contract.minTemperature) {
                penalty += (contract.minTemperature - lowestReading.centigrade) * contract.minPenaltyFactor;
                console.log('Min temp penalty: ' + penalty);
            }
```

Slika 5.7. Dio asinkrona funkcije za primljenu robu

```
            // provjeravamo je li max temperatura krši ugovor
            if (highestReading.centigrade > contract.maxTemperature) {
                penalty += (highestReading.centigrade - contract.maxTemperature) * contract.maxPenaltyFactor;
                console.log('Max temp penalty: ' + penalty);
            }

            // dodavanje kazni
            payOut -= (penalty * shipment.unitCount);

            if (payOut < 0) {
                payOut = 0;
            }
        }
    }

    console.log('Payout: ' + payOut);
    contract.grower.accountBalance += payOut;
    contract.importer.accountBalance -= payOut;

    console.log('Grower: ' + contract.grower.$identifier + ' new balance: ' + contract.grower.accountBalance);
    console.log('Importer: ' + contract.importer.$identifier + ' new balance: ' + contract.importer.accountBalance);

    // update proizvođačevog računa
    const growerRegistry = await getParticipantRegistry('org.acme.shipping.perishable.Grower');
    await growerRegistry.update(contract.grower);

    // update uvoznikovog računa
    const importerRegistry = await getParticipantRegistry('org.acme.shipping.perishable.Importer');
    await importerRegistry.update(contract.importer);

    // update pošiljke
    const shipmentRegistry = await getAssetRegistry('org.acme.shipping.perishable.Shipment');
    await shipmentRegistry.update(shipment);
}
```

Slika 5.8. Nastavak asinkrone funkcije za primljenu robu

Na slici 5.9 prikazana je asinkrona funkcija *temperatureReading()* koja prima parametar s čitača temperature proizvoda. Funkcija provjerava postoji li već neka vrijednost unutar pošiljke, s obzirom da svaki paket ne zahtjeva čitač temperature. Ukoliko postoji, primljenu temperaturu stavlja na kraj polja (eng. *array*), a ukoliko ne postoji onda stavlja primljenu temperaturu kao prvi element polja. Nakon toga, nove podatke ažurira na pošiljku.

```
/**
 * Očitana temperatura je primljena od prijevoznika
 * @param {org.acme.shipping.perishable.TemperatureReading} temperatureReading - the TemperatureReading transakcija
 * @transaction
 */
async function temperatureReading(temperatureReading) {

    const shipment = temperatureReading.shipment;

    console.log('Adding temperature ' + temperatureReading.centigrade + ' to shipment ' + shipment.$identifier);

    if (shipment.temperatureReadings) {
        shipment.temperatureReadings.push(temperatureReading);
    } else {
        shipment.temperatureReadings = [temperatureReading];
    }

    // dodavanje čitača temperature pošiljci
    const shipmentRegistry = await getAssetRegistry('org.acme.shipping.perishable.Shipment');
    await shipmentRegistry.update(shipment);
}
```

Slika 5.9. Asinkrona funkcija za očitavanje temperature

Slike 5.10 i 5.11 prikazuju implementaciju asinkrone funkcije *setupDemo()* koja služi kao transakcija pomoću koje se *uploda* predefinirani sustav na lanac blokova. Korištena je metoda *getFactory()*, koja služi za kreiranje instanci objekata kao što su *participant*, *asset* i *transaction*, metodu *newResource()* za kreiranje novih instanci spomenutih objekata te metodu *newConcept()* pomoću koje se kreira novi tip podatka *concept*, u ovom slučaju novu adresa. Također, korištena je metoda *newRelationship()* koja služi kao pokazivač prema već postojećoj instanci, odnosno kako bi povezali proizvođača, prijevoznika i uvoznika u jedan ugovor. Također se postavlja vrijeme isporuke na jedan dan, zatim određene vrijednosti za proizvod, minimalnu i maksimalnu temperaturu na kojoj se taj proizvod mora prevoziti te se stavljaju unutar ugovora. Nakon toga je prikazan proces kreiranja pošiljke koji se povezuje s ugovorom. Na kraju su prikazani primjeri kreiranja novih korisnika, ugovora i pošiljke pri čemu se koristi procedura *await* koja čeka na rješenje objekta *Promise*, a on predstavlja uspješni dovršetak asinkrone funkcije.

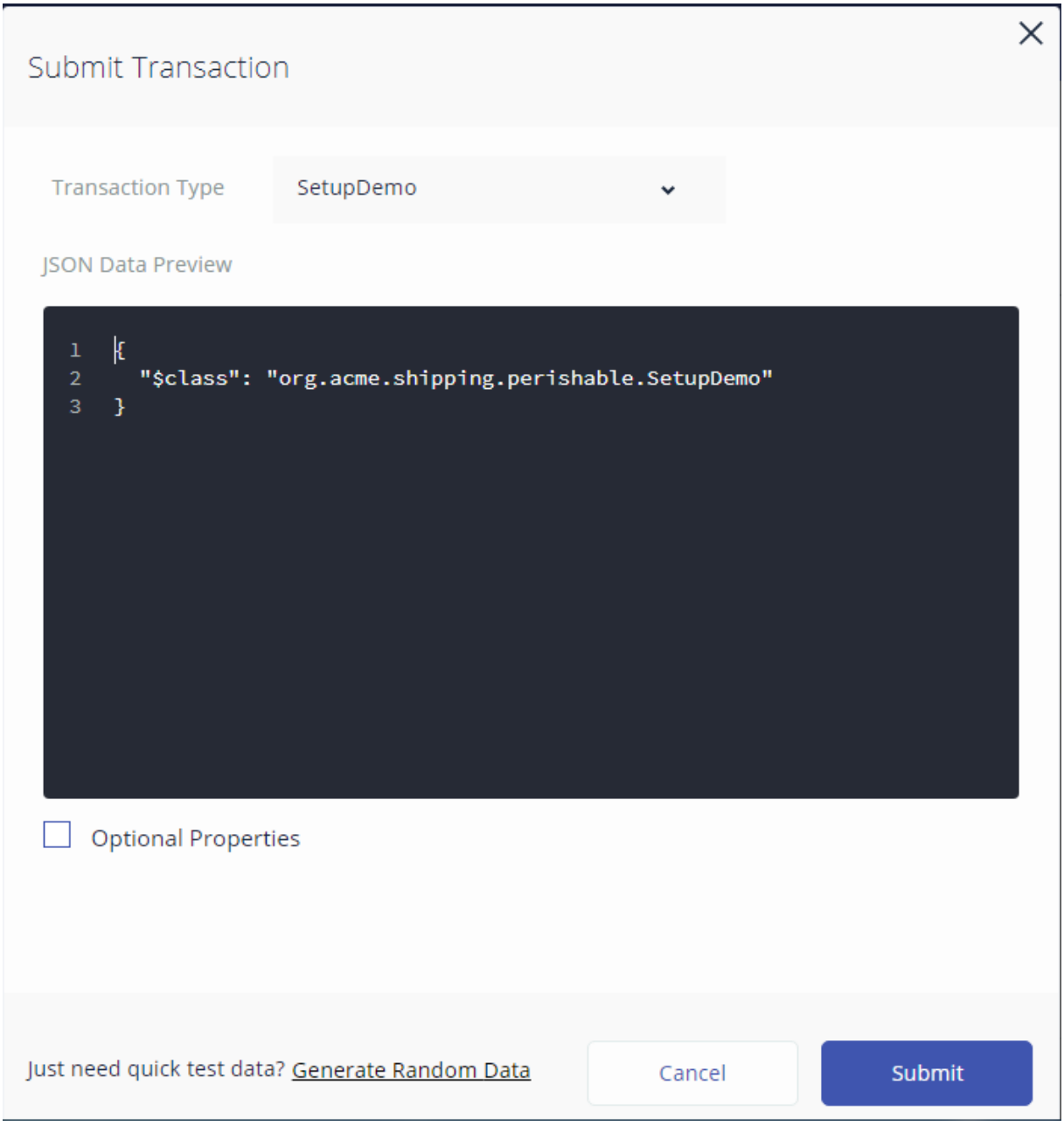
```
/**
 * Stvaranje testova, imovine i korisnika za pokretanje rada.
 * @param {org.acme.shipping.perishable.SetupDemo} setupDemo - the SetupDemo transakcija
 * @transaction
 */
async function setupDemo(setupDemo) {

    const factory = getFactory();
    const NS = 'org.acme.shipping.perishable';

    // stvaranje proizvođača
    const grower = factory.newResource(NS, 'Grower', 'farmer@email.com');
    const growerAddress = factory.newConcept(NS, 'Address');
    growerAddress.country = 'CRO';
    grower.address = growerAddress;
    grower.accountBalance = 0;

    // stvaranje uvoznika
    const importer = factory.newResource(NS, 'Importer', 'supermarket@email.com');
    const importerAddress = factory.newConcept(NS, 'Address');
    importerAddress.country = 'FR';
    importer.address = importerAddress;
    importer.accountBalance = 0;

    // stvaranje prijevoznika
    const shipper = factory.newResource(NS, 'Shipper', 'shipper@email.com');
    const shipperAddress = factory.newConcept(NS, 'Address');
    shipperAddress.country = 'UK';
    shipper.address = shipperAddress;
    shipper.accountBalance = 0;

    // stvaranje ugovora
    const contract = factory.newResource(NS, 'Contract', 'CON_001');
    contract.grower = factory.newRelationship(NS, 'Grower', 'farmer@email.com');
    contract.importer = factory.newRelationship(NS, 'Importer', 'supermarket@email.com');
    contract.shipper = factory.newRelationship(NS, 'Shipper', 'shipper@email.com');
    const tomorrow = setupDemo.timestamp;
    tomorrow.setDate(tomorrow.getDate() + 1);
    contract.arrivalDateTime = tomorrow; // pošiljka mora stići sutra
    contract.unitPrice = 0.5; // 50lipa po proizvodu
```

Slika 5.10. Asinkrona funkcija za stvaranje proizvođača, uvoznika, prijevoznika i ugovora

```
    contract.minTemperature = 2; // min temperatura za proizvod
    contract.maxTemperature = 10; // max temperatura za proizvod
    contract.minPenaltyFactor = 0.2; // smanjujemo cijenu za 20 lipa za svaki stupanj ispod min temp
    contract.maxPenaltyFactor = 0.1; // smanjujemo cijenu za 10 lipa za svaki stupanj iznad max temp

    // stvaranje pošiljke
    const shipment = factory.newResource(NS, 'Shipment', 'SHIP_001');
    shipment.type = 'BANANAS';
    shipment.status = 'IN_TRANSIT';
    shipment.unitCount = 5000;
    shipment.contract = factory.newRelationship(NS, 'Contract', 'CON_001');

    // dodavanje proizvođača
    const growerRegistry = await getParticipantRegistry(NS + '.Grower');
    await growerRegistry.addAll([grower]);

    // dodavanje uvoznika
    const importerRegistry = await getParticipantRegistry(NS + '.Importer');
    await importerRegistry.addAll([importer]);

    // dodavanje prijevoznika
    const shipperRegistry = await getParticipantRegistry(NS + '.Shipper');
    await shipperRegistry.addAll([shipper]);

    // dodavanje ugovora
    const contractRegistry = await getAssetRegistry(NS + '.Contract');
    await contractRegistry.addAll([contract]);

    // dodavanje pošiljke
    const shipmentRegistry = await getAssetRegistry(NS + '.Shipment');
    await shipmentRegistry.addAll([shipment]);
}
```

Slika 5.11. Ostatak asinkrone funkcije za stvaranje pošiljke te dodavanje korisnika, ugovora te pošiljaka

Sljedeći korak u izradi ovog primjera je implementacija napisanog koda u Hyperledger lancu blokova. Hyperledger je *open source* Linuxova fondacija za razvoj tehnologije lanca blokova s velikim broje Fortune 500 tvrtki koji su članovi. Pri izradi ovog primjera korišten je Hyperledger Fabric platforma koja omogućava razvoj poslovnog lanca blokova razvijenog od strane IBM-a. Već spomenuti kod napisan u VSCodu pretvoren je u .bna format te *uploadan* u Hyperledger lanac blokova. Prvi korak pri implementacije je pokrenuti *SetupDemo* transakciju kako bi se učitali predefinirani objekti. Postupak pokretanja *SetupDemo* transakcije je prikazan na slici 5.12. Instance predefiniranih objekata prikazane su na slikama 5.13, 5.14, 5.15, 5.16, 5.17.



Slika 5.12. Pokretanje SetupDemo transakcije

ID	Data

farmer@email.com	{
	"$class": "org.acme.shipping.perishable.Grower",
	"email": "farmer@email.com",
	"address": {
	"$class": "org.acme.shipping.perishable.Address",
	"country": "CRO"
	},
	"accountBalance": 5000
	}

Slika 5.13. Kreirani proizvođač

ID	Data

supermarket@email.com	{
	"$class": "org.acme.shipping.perishable.Importer",
	"email": "supermarket@email.com",
	"address": {
	"$class": "org.acme.shipping.perishable.Address",
	"country": "FR"
	},
	"accountBalance": 5000
	}

Slika 5.14. Kreirani uvoznik

ID	Data

shipper@email.com	{
	"$class": "org.acme.shipping.perishable.Shipper",
	"email": "shipper@email.com",
	"address": {
	"$class": "org.acme.shipping.perishable.Address",
	"country": "UK"
	},
	"accountBalance": 5000
	}

Slika 5.15. Kreirani prijevoznik

ID	Data

CON_001	{
	"$class": "org.acme.shipping.perishable.Contract",
	"contractId": "CON_001",
	"grower": "resource:org.acme.shipping.perishable.Grower#farmer@email.com",
	"shipper": "resource:org.acme.shipping.perishable.Shipper#shipper@email.com",
	"importer": "resource:org.acme.shipping.perishable.Importer#supermarket@email.com",
	"arrivalDateTime": "2018-09-13T17:48:48.948Z",
	"unitPrice": 0.5,
	"minTemperature": 2,
	"maxTemperature": 10,
	"minPenaltyFactor": 0.2,
	"maxPenaltyFactor": 0.1
	}

Slika 5.16. Kreirani ugovor

ID

Data

SHIP_001


```
{
  "$class": "org.acme.shipping.perishable.Shipment",
  "shipmentId": "SHIP_001",
  "type": "BANANAS",
  "status": "IN_TRANSIT",
  "unitCount": 5000,
  "contract": "resource:org.acme.shipping.perishable.Contract#CON_001"
}
```


Slika 5.17. Kreirana pošiljka

Kao što se vidi na slici 5.17 u redu „status“ pošiljka se još prevozi. Ukoliko uvoznik pokrene proceduru da je pošiljka stigla, „status“ se mijenja što se vidi na slici 5.18 i tako završava proces izmjene dobara između različitih poduzetnika.

ID

Data

SHIP_001


```
{
  "$class": "org.acme.shipping.perishable.Shipment",
  "shipmentId": "SHIP_001",
  "type": "BANANAS",
  "status": "ARRIVED",
  "unitCount": 5000,
  "temperatureReadings": [
    {
      "$class": "org.acme.shipping.perishable.TemperatureReading",
      "centigrade": 5,
      "shipment": "resource:org.acme.shipping.perishable.Shipment#SHIP_001",
      "transactionId": "1362fe9c-6770-4d81-84b2-46834bd45405",
      "timestamp": "2018-09-12T18:48:30.871Z"
    }
  ],
  "contract": "resource:org.acme.shipping.perishable.Contract#CON_001"
}
```


Slika 5.18. Prikaz promjene statusa

Ovaj primjer prikazuje efikasno korištenje tehnologije lanca blokova koja je omogućila sigurnu razmjenu dobara putem pametnih ugovora. Njime je dokazano da tehnologija lanca blokova nije korisna isključivo za kreiranje i korištenje kriptovaluta već primjenjiva i u ostalim granama industrije.

6. ZAKLJUČAK

Lanac blokova je relativno nova tehnologija koja omogućuje siguran prijenos informacija između dviju stranaka bez posredovanja treće, brže i sigurnije globalne transakcije te nove načine za rješavanje postojećih problema. Iako je ova tehnologija postala poznata u svijetu zbog popularnosti kriptovaluta, to joj nije jedina primjena. Ona je također tehnološka osnova za razvitak pametnih ugovora širokog spektra primjenjivosti što je dokazano praktičnim primjerom. Iz toga se može zaključiti da je lanac blokova tehnologija koja donosi veliki potencijal za unapređivanje globalne ekonomije i načina poslovanja. Usprkos prednostima ona nije savršena, jer zahtjeva velike količine fizičkih spremnika. Oni nastaju zbog potrebe da se podaci kopiraju velik broj puta čime se stvara obilje bespotrebnih podataka. Unatoč nedostacima velika je vjerojatnost da će se ova tehnologija, koja nije još uvijek ispunila svoj puni potencijal, u budućnosti implementirati u različita područja, osobito u područje bankarstva i *online* transakcija.

Ovaj završni rad prikazuje praktični primjer poslovne mreže koja se bavi razmjenom dobara između proizvođača i uvoznika. Napravljen je u Visual Studio Code editoru te implementiran u Hyperledger lanac blokova. Analizom praktičnog primjera pokazano je da se svaka transakcija zapisuje u lanac blokova kako bi se spriječilo varanje sustava, što se postiže primjenom pametnih ugovora koji definiraju poslovni odnos između korisnika. Iz toga se može zaključiti da tehnologija lanca blokova pruža sigurnu okolinu za razmjenu sredstava i u slučajevima kada ne postoji povjerenje između stranaka.

LITERATURA

[1] M. Gupta, Blockchain for Dummies, IBM, John Wiley & Sons, Inc., Hoboken, New York, 2017

[2] A. Singh, What is the history of blockchain technology: The journey of Blockchain started in 1991, Quora, 2018, dostupno na: https://www.quora.com/What-is-the-history-of-blockchain-technology [26. lipnja 2018.]

[3] Blockchain: History, Wikipedia, 2018, dostupno na: https://en.wikipedia.org/wiki/Blockchain#History [26. lipnja 2018.]

[4] Blockchain: History, Wikipedia, 2018 dostupno na: https://en.wikipedia.org/wiki/Blockchain#History [26. lipnja 2018.]

[5] G. Konstantopoulos, Understanding Blockchain Fundamentals, Part 2: Proof of Work & Proof of Stake, A Medium Corporation, 2017, dostupno na: https://medium.com/loom-network/understanding-blockchain-fundamentals-part-2-proof-of-work-proof-of-stake-b6ae907c7edb [20. lipnja 2018.]

[6] G. Konstantopoulos, Understanding Blockchain Fundamentals, Part 2: Proof of Work & Proof of Stake, A Medium Corporation, 2017, dostupno na: https://medium.com/loom-network/understanding-blockchain-fundamentals-part-2-proof-of-work-proof-of-stake-b6ae907c7edb [20. lipanj 2018.]

[7] M. Gupta, Blockchain for Dummies, IBM, John Wiley & Sons, Inc., Hoboken, New York, 2017.

[8] L.S. Stephens, Explain Delegated Proof of Stake Like I'm 5: What are the ingredients for DPOS, Hackernoon, 2017, dostupno na: https://hackernoon.com/explain-delegated-proof-of-stake-like-im-5-888b2a74897d [22. lipnja 2018.]

[9] D. Drescher, Blockchain Basic – A Non-Tehnical Introduction in 25 Steps, Apress, Frankfurt am Main, Germany, 2017.

[10] Block, Bitcoin Wiki, dostupno na: https://en.bitcoin.it/wiki/Block [12.rujna 2018.]

[11] Block Header (Cryptocurrency), Investopedia, LLC., dostupno na: https://www.investopedia.com/terms/b/block-header-cryptocurrency.asp [12. rujna 2018.]

[12] A. Brownworth, Blockchain Demo: Coinbase, dostupno na: https://anders.com/blockchain/coinbase.html [25. lipnja 2018.]

[13] D. Drescher, Blockchain Basic – A Non-Tehnical Introduction in 25 Steps, Apress, Frankfurt am Main, Germany, 2017.

[14] A. Robbins, Blockchain Fundamentals #1: What is a Merkle Tree?, A Medium Corporation, 2018, dostupno na: https://medium.com/byzantine-studio/blockchain-fundamentals-what-is-a-merkle-tree-d44c529391d7 [15. rujna 2018.]

[15] A. Prabhakar, BlockChain Fundamentals Part 1: What is Hashing, 2017, dostupno na: https://ajithp.com/tag/merkle-tree/ [27. lipnja 2018.]

[16] A. Ho, How can we generate a Merkle Tree with an odd number of leaves or an odd number of parent nodes, Quora, 2018, dostupno na: https://www.quora.com/How-can-we-generate-a-Merkle-Tree-with-an-odd-number-of-leaves-or-an-odd-number-of-parent-nodes-say-5-leaves-as-an-example [27. lipnja 2018.]

[17] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System: Reclaiming Disk Space, Learn Cryptography, 2008, dostupno na: https://learncryptography.com/cryptocurrency/the-bitcoin-whitepaper [28. lipnja 2018.]

[18] CoinMarketCap, dostupno na: https://coinmarketcap.com/all/views/all/ [4. rujan 2018.]

[19] B. Marr, A Short History of Bitcoin And Crypto Currency Everyone Should Read: 2008 – The Mysterious Mr Nakamoto, Forbes Media LLC., 2018, dostupno na: https://www.forbes.com/sites/bernardmarr/2017/12/06/a-short-history-of-bitcoin-and-crypto-currency-everyone-should-read/#49cec9813f27 [16. rujna 2018.]

[20] E. Cheng, There are now 17 million bitcoins in existence – only 4 million left to „mine“, CNBC, 2018, dostupno na: https://www.cnbc.com/2018/04/26/there-are-now-17-million-bitcoins-in-existence--only-4-million-left-to-mine.html [4. rujan 2018.]

[21] What is Ethereum? A Stem-by-Step Beginners Guide, Blockgeeks Inc., 2018, dostupno na: https://blockgeeks.com/guides/ethereum/ [16. rujna 2018.]

[22] B. Asolo, What is VeChain?, MyCryptoPedia, 2018, dostupno na: https://www.mycryptopedia.com/vechain-ven-explained/ [16. rujna 2018.]

[23] I. Voras, Što su pametni ugovori – uvod, UBIK, 2018, dostupno na: https://ubik.hr/2018/03/26/sto-su-pametni-ugovori-uvod/ [16. rujna 2018.]

[24] Smart Contracts: The Blockchain Technology That Will Replace Lawyers: What are Smart Contracts, Blockgeeks Inc., 2016, dostupno na: https://blockgeeks.com/guides/smart-contracts/ [26. lipnja 2018.]

[25] L. Goasduff, S.C., Hippold, Gartner Survey Reveals the Scarcity of Current Blockchain Deployments: Blockchain Adoption, Worldwide, Gartner Inc., Egham, UK, 2018, dostupno na: https://www.gartner.com/newsroom/id/3873790 [16. rujna 2018.]

[26] A. Ross, A guide to blockchain-as-a-service (BaaS) for CTOs and IT leaders: Blockchain-as-a-service, Bonhill Group, 2018, dostpuno na: https://www.information-age.com/essential-guide-blockchain-as-a-service-123473581/ [16. rujna 2018.]

[27] M. Gord, How to Get Started with the Embark Framework, MGL Blockchain Consulting, 2018, dostupno na: https://mlgblockchain.com/intro-embark.html [15. rujna 2018.]

[28] A. Lin, Solidity Code Compilation, A Medium Corporation, 2018, dostupno na: https://medium.com/coinmonks/solidity-code-compilation-334dc3b3784e [16. rujna 2018.]

[29] Liaison Partners With Tierion To Secure Enterprise Data Using Blockchain Proof Technology, A Medium Corporation, 2018, dostupno na: https://medium.com/tierion/liasion-partners-with-tierion-to-secure-enterprise-data-using-blockchain-proof-technology-d7001e6ad506 [15. rujna 2018.]

SAŽETAK

Ovaj završni rad prikazuje teorijsku analizu tehnologije lanca blokova te prikaz primjene te tehnologije na praktičnom primjeru. Teorijski obrađuje pojmove lanca blokova, njegovu povijest, konsenzuse i neke od njegovih metoda, strukturu blokova te kriptografiju unutar lanca blokova. Također, prikazuje primjere korištenja tehnologije lanca blokova poput pametnih ugovora i kriptovaluta te pokazuje da kriptovalute nisu istovjetne lancu blokova već da je on samostalna tehnologija koja se može implementirati u razne sustave poput spomenutih pametnih ugovora. Zbog velikih ulaganja u kreiranje vlastitih lanaca blokova, iskorištenost tehnologije je vrlo mala s obzirom na njen potencijal. Stoga su stvorene razne okoline i alati za jednostavnije i jeftinije korištenje te tehnologije. Upravo korištenjem jednog od alata, zvanog BaaS, stvoren je praktični primjer decentralizirane aplikacije poslovne mreže. Aplikacija je razvijena pomoću Visual Studio Code editora i Hyperledger lanca blokova, a iz njene izrade se može vidjeti primjena ove tehnologije u realnom industrijskom sustavu.

Ključne riječi: decentralizirana aplikacija, konsenzus, kriptografija, kriptovalute, lanac blokova, pametni ugovori

ABSTRACT

This paper theoreticall analyse blockchain technology and shows practical use of that technology. Theoretically it describes blockchain technology, history of blockchain, consensus and some of his methods, structure of a block in a blockchain and the use of cryptography in a blockchain. Also, practical uses of blockchain technology like smart contracts and cryptocurrencys are shown to prove that cryptocurrency and blockchain are not the same. It is used to show that blockchain is a standalone technology which can be implemented in different types of systems like aformentioned smart contracts. Since large investments are needed for creating personal blockchains, the use of this technology is very small considering her potential. Therefore, a lot of environments and tools have been created to allow easier and cheaper blockchain development, one of which has been used for developing this decentralized business network application. The tool used is called BaaS or Blockchain as a Service. Application, which shows practical use of this technology in a real industrial system, has been developed using Visual Studio Code editor and Hyperledger blockchain.

Keywords: blockchain, consensus, cryptocurrency, cryptography, decentralize application, smart contracts

ŽIVOTOPIS

Lovro Pejčić rođen je 13. veljače 1996. u Osijeku. U Osijeku završava osnovnu školu „Dobriša Cesarić“ te 2011. upisuje Jezičnu gimnaziju. 2015. godine ostvaruje upis na Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, smjer računarstvo. Od 1. veljače do 1. ožujka 2018. radi kao praktikant u Atos Convergence Creators u Osijeku.

PRILOZI

Prilog 1. Dokument završnog rada

Prilog 2. Pdf završnog rada

Prilog 3. Primjeri i programski kodovi