SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Neven Travaš

PRIMJENA BLOCKCHAINA I PAMETNIH UGOVORA

ZAVRŠNI RAD

Varaždin, 2018.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Neven Travaš

Matični broj: 44049/15–R

Studij: Informacijski sustavi

PRIMJENA BLOCKCHAINA I PAMETNIH UGOVORA

ZAVRŠNI RAD

Mentor:

Doc. dr. sc. Tonimir Kišasondi

Varaždin, srpanj 2018.

Neven Travaš

Izjava o izvornosti

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

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Rad se bavi proučavanjem i analizom blockchaina kao mehanizma distribuiranog zapisa podataka koji čuva integritet na distribuiranoj razini i kod kojeg nije potrebno imati povjerenje između sudionika u sustavu. Ukratko pojašnjava pojam blockchaina, pametnih ugovora, distribuiranih sustava i decentraliziranih aplikacija. Prikazuje izradu lokalnog blockchaina na kojem dvije stranke mogu međusobno komunicirati i izvršavanje jedne decentralizirane aplikacije (DAPP) pomoću pametnih ugovora na Hyperledger blockchain arhitekturi.

Ključne riječi: blockchain, pametni ugovori, Hyperledger.

Sadržaj

1. Uvod

Ugovori i transakcije, njihovo nadgledavanje i skladištenje, definiraju strukturu svjetskog poslovanja bilo u pravnom, političkom ili ekonomskom smislu. I dok su u današnje vrijeme, zahvaljujući digitalizaciji, informatizaciji i globalizaciji ljudi u stanju olakšati si svakodnevni život koristeći se raznim Internet servisima koje im pružaju svakojake kompanije, birokracijski procesi provođenja i skladištenja ugovora ili provjeravanja i provođenja transakcija i dalje predstavljaju dugotrajne i mukotrpne procese. Korištenjem tuđih servisa i usluga korisnici istovremeno ovise o načinu na koji kompanije koriste njihove podatke, odnosno moraju imati povjerenja u centralizirane sustave i razne treće stranke kako bi uživali u luksuzu njihovog proizvoda. Iz istog razloga potpuna digitalizacija ugovora ili provedenih transakcija smatrala se nemogućom pošto se nikako nije moglo postići potpuno povjerenje između dvije poslujuće stranke. Kako bi se stalo na kraj problemu povjerenja a pritom ubrzao i pojednostavio proces transakcija i poslovanja javlja se koncept blockchaina. Distribuiranog zapisa podataka koji čuva integritet na distribuiranoj razini i kod kojeg nije potrebno imati povjerenje između sudionika u sustavu. Jedan od ideja kako pojednostaviti i ubrzati transakcije bez utjecaja treće stranke koja garantira povjerenje je pametni ugovor (eng. Smart contract). Vrsta programa (ugovora) koji se pokreće na blockchainu i omogućuje provođenje poslovnih pravila kroz programski kod. Upotreba blockchaina kao tehnologije na kojoj se provode pametni ugovori moglo bi ubrzati i pojednostaviti transakcije u svim područjima poslovanja, te pojednostaviti proces određivanja autentičnosti i vlasništva.

2. Metode i tehnike rada

Prilikom razrade teme koriste se metode analize i sinteze, metoda apstrakcije i konkretizacije. Od programskih alata korišteni su programski jezici: Go i Javascript, node packet manager (npm) i javno dostupni Hyperledger fabric git repozitorij. Analizom blockchain tehnologije čitatelju će se pojasniti koncept blockchaina, pametnih ugovora i decentraliziranih aplikacija popraćene primjerima provedenim na Linux operacijskom sustavu.

3. Distribuirani sustavi

Kako bi shvatili pojam blockchain-a potrebno je shvatiti pojam distribuiranih sustava, na kojima se zapravo zasniva blockchain tehnologija.

Distribuirani sustav oblik je računalnog sustava u kojem dva ili više čvora međusobno komuniciraju kako bi ostvarili neki zajednički cilj. Sustav je modeliran na način da ga krajnji korisnici vide kao jednu logičku platformu. Čvor je definiran kao jedinica unutar sustava (npr. Jedno računalo). Svi čvorovi unutar sustava mogu međusobno slati i primati poruke, a svaki čvor u sustavu može biti ispravan, neispravan ili maliciozan te svaki čvor ima svoju memoriju i procesor. Glavni izazov u izgradnji distribuiranog sustava je komunikacija između čvorova i postupanje s kvarovima čvorova. Odnosno, iako neki čvorovi možda zakažu ili postanu maliciozn sustav bi i dalje trebao ispravno raditi sa svim preostalim čvorovima. Glavne karakteristike koje bi morao zadovoljavati svaki distribuirani sustav su konzistentnost (svaki čvor u sustavu raspolaže sa najstarijom verzijom podataka), pristupačnost (sustav je aktivan i spreman za korištenje, bez grešaka u prijenosu za svo vrijeme rada) i tolerancija na promjene (osigurava da ako dio čvorova zakaže sustav i dalje normalno funkcionira i radi sa čvorovima koji su ispravni). S glavnim karakteristikama na umu razvija se oblik distribuiranog sustava koji je dobio naziv blockchain.

Blockchain možemo promatrati kao oblik ne promjenjive, ali nadogradive baze podataka koja se replicira sinkronizirano na sve čvorove postojećeg distribuiranog sustava, koja podržava praćenje događaja (eng. Event sourcing), što nam omogućava rekreaciju nekog prijašnjeg stanja sustava. Na temelju takvog jednog sustava dalje se razvio blockchain. (Kozlovski, 2018.; Bashir, 2017.)

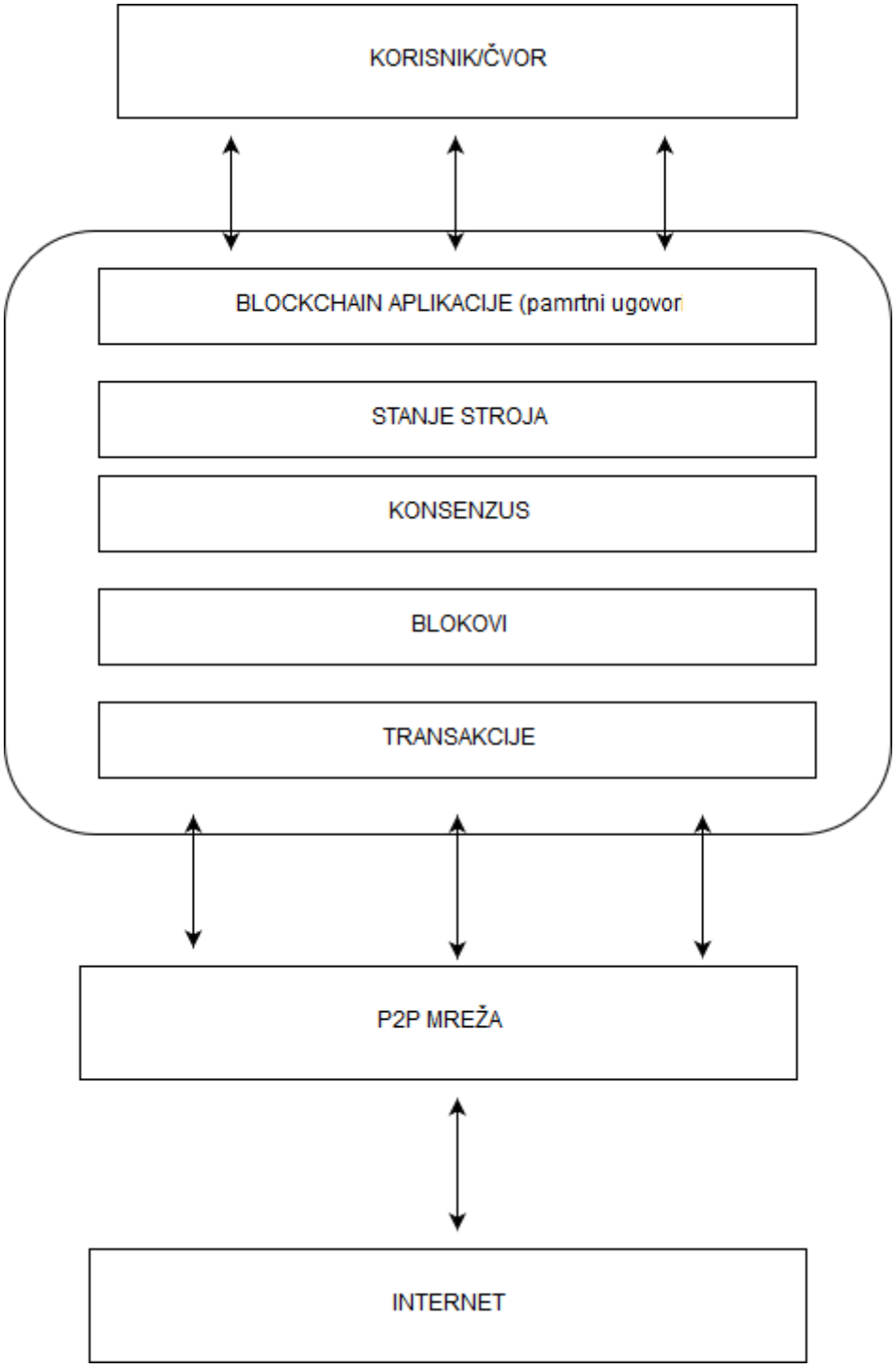
4. Blockchain

Blockchain prvi puta je predstavljen 2008. godine s izumom Bitcoina, a prvi puta implementiran 2009. godine. Iako se rad ne bavi Bitcoinom kao digitalnom valutom ista će biti ukratko spomenuta i objašnjena pošto bez Bitcoina ne bi bilo blockchaina. Bitcoin je decentralizirana, anonimna i distribuirana platna mreža a ujedno i digitalna valuta koju platna mreža koristi. Temelji se na Proof of work (PoW) konceptu pri kojem je za stvaranje nove količine valute ili provedbu transakcije potrebno riješiti kompleksan matematički problem koji se verificira od strane sustava i onda odobrava.

Sam sustav na kojem se bitcoin (u ovom slučaju valuta) vrti i preko kojeg se odrađuju transakcije i stvaraju novi bitcoini zove se blockchain, i ovisno da li se gleda s poslovne ili tehničke strane može biti interpretiran na više načina.

U svojem korijenu blockchain čini tako zvana javna knjiga (eng. Distributed ledger) koja je umrežena sa svakim čvorom u sustavu konceptom svaki sa svakim (eng. peer to peer), koja je kriptografski zaštićena i nepromjenjiva (ili veoma teško promjenjiva), na koju se mogu samo nadodati nove transakcije ili čitati stare, a svaka promjena mora bit odobrena konsenzusom ili dogovorom između čvorova.

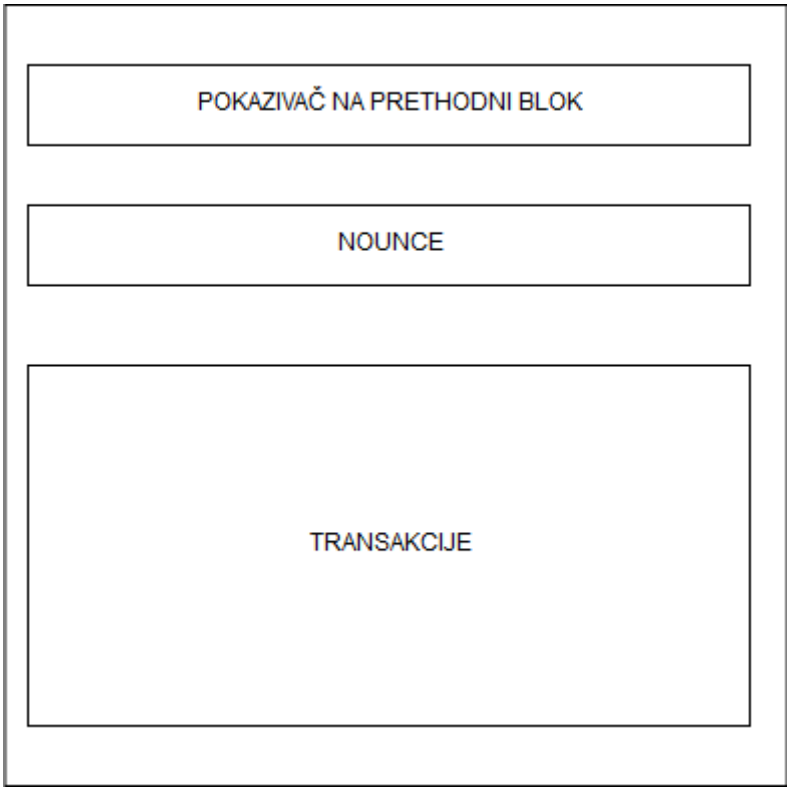
Možemo ga promatrati kao sloj distribuirane mreže čvorova međusobno povezanih svaki sa svakim, koji se odvija povrh Interneta. (Bashir, 2017., Antonopoulous 2017.)



Slika 1 Mrežni prikaz blockchaina (izvor: Bashir, 2017)

S poslovnog stajališta blockchain možemo promatrati kao platformu na kojoj čvorovi mogu izmjenjivati vrijednosti preko transakcija bez potrebe centralnog autoriteta koji bi osiguravao ispravnost transakcije.

Sadržano u samom imenu, blockchain čini skupina „blokova“ međusobno povezanih u lanac. Možemo ga usporediti s vezanom listom. Svaki blok čini logičku organizaciju provedenih transakcija, a veličina bloka ovisi o načinu implementacije blockchaina. Svaki blok uključuje i heširani pokazivač na prijašnji blok u lancu, osim ako se radi o prvom bloku u lancu (eng. Genesis block). Struktura bloka također ovisi o tipu blockchaina i načinu implementacije, a glavni atributi koje svaki blok mora sadržavati su: zaglavlje, pokazivač na prijašnji blok, vremenska oznaka, nonce (nasumičan broj koji se koristi pri izračunavanju novog bloka), brojač transakcija i transakcije. (Bashir, 2017.)



Slika 2 Struktura bloka (Prema: Bashir,2017)

4.1. Opći elementi blockchaina

U opće elemente blockchaina spadaju adrese, transakcije, blokovi, mreža na kojoj se provodi, skriptni ili programski jezik, virtualni stroj, čvor i pametni ugovor.

Adrese su jedinstveni identifikatori koje se koriste prilikom transakcija za određivanje pošiljatelja i primatelja. Adresa je reprezentirana javnim ključem. Korisnik može koristiti istu adresu za više transakcija, ali je adresa sama po sebi jedinstvena i ne postoje dvije iste adrese. Međutim korisnik ne može zadržati istu adresu i generirati novu prilikom nove transakcije. Odnosno u praksi svaki korisnik pošiljatelj prilikom svake nove transakcije generira novu adresu s koje šalje.

Transakcija je temeljna jedinica svakog blockchaina i predstavlja transfer vrijednosti između korisnika, odnosno između dvije adrese. Svaka transakcija zapisana je na bloku unutar blockchaina, a svaki blok sadrži vezu na prijašnji blok i nonce. Što se tiče mrežne topologije blockchain se odvija na Peer to peer mrežnoj arhitekturi u kojoj svaki sudionik mreže može komunicirati, slati i primati poruke od ostalih sudionika u mreži.

Skriptni ili programski jezici omogućuju provedbu operacija nad transakcijama. Transakcijske skripte čine predefinirani set naredbi prema kojima čvorovi vrše transakcije s jedne na drugu adresu.

Virtualni stroj (VM – od engleske riječi Virtual machine), čini ekstenziju transakcijske skripte. Omogućava izvođenje programskog koda na blockchainu u obliku pametnih ugovora. Pošto same transakcijske skripte mogu biti limitirane brojem mogućih operacija. Pametni ugovori su programi koji se izvršavaju na blockchainu, sadrže poslovnu logiku koja se provede kada su zadovoljeni potrebni uvjeti za njihovo izvršavanje. Pametni ugovori nisu implementirani na svim sadašnjim blockchainovima, međutim dobivaju na popularnosti zbog fleksibilnosti koju omogućuju u kontroli aplikacija koje se izvršavaju na blockchainu. Više o pametnim ugovorima u sljedećim poglavljima.

Sam blockchain možemo promatrati kao konačni automat koji se sastoji od više čvorova, u kojem se stanje mijenja od početnog do završnog kao rezultat uspješno provedenih transakcija između čvorova. Čvor u sklopu blockchaina provodi razne funkcije ovisno o njegovoj ulozi unutar mreže. Čvor može provesti ili odobriti određenu transakciju te se koristiti se za rudarenje. Na taj način olakšava održavanje koncensusa blockchaina. (Bashir, 2017.)

4.2. Tipovi blockchaina

Kroz godine blockchain se razvio u više smjerova, zbog čega su se razvili više tipova blockchaina sa različitim ali i često sličnim karakteristikama. Razlikujemo: javni, privatni i konzorcijski ili federalni blockchain.

Javni blockchain, otvoren je javnosti i unutar njega svatko može sudjelovati kao čvor u donošenju odluka, a korisnici mogu ali ne moraju biti nagrađeni za njihov doprinos. Svaki korisnik ima kopiju javne knjige (eng. ledger) na svom lokalnom čvoru i koristi distribuirani mehanizam konsenzusa kako bi se donijela odluka o daljnjem stanju blockchaina.

Privatni blockchain, kao što i samo ime kaže, je privatan i otvoren samo određenoj grupi ljudi ili organizacijama koje su odlučili svoje transakcije vršiti putem blockchaina.

Konzorcijski ili federalni blockchain, sličan je privatnom blockchainu samo umjesto da je jedan čvor glavni, više ih je glavnih. Radi se o skupini predstavnika ili poduzeća koji donose najvažnije odluke za benefit cijele mreže. (Bashir, 2017.)

4.3. Korist i ograničenja blockchaina

Od pojave blockchaina kao koncepta, njegove prve interpretacije pa do danas, raspravlja se o mnogim koristima blockchain tehnologije, neke od glavnih čine decentralizaija, transparentnost i povjerenje, nepromjenjivost, visoka pristupačnost, sigurnost, brzina transakcije te manji trošak. Decentralizacija jedan je od glavnih koncepata na kojima se bazira blockchain. Glavna ideja je da nije potrebna treća stranka koja osigurava korektnost transakcije, umjesto toga implementira se mehanizam konsenzusa kako bi se osigurala korektnost transakcije. Najčešći oblici mehanizma konsenzusa su Proof of work (PoW) i Proof of Stake (PoS).

Proof of Work (PoW) tip je konsenzusa koji se bazira na tome da se utrošilo mnogo računalnih resursa prije nego se prezentira vrijednost ili rješenje koje se onda prihvaća od sustava. Smatra se dobrim rješenjem pošto je trenutno jedini algoritam koji se pokazao uspješnim protiv Sybil napada, a funkcionira na principu rješavanja kompleksnih matematičkih jednadžbi čije se rješenje verificira s mrežom i ukoliko je prihvaćeno transakcija se prihvaća u blok.

Proof of Stake (PoS) tip je konsenzusa koji funkcionira na ideji da korisnik/čvor ima dovoljan ulog unutar mreže da se korisniku ne isplati napasti mrežu, odnosno transakciju odobravaju oni čvorovi koji imaju najveći ulog unutar mreže.

Delegated Proof of Stake (DPOS) nadodaje se na klasični PoS na način da korisnici koji imaju najveće udjele u mreži glasovanjem mogu izabrati da li će određena transakcija biti prihvaćena.

Podaci s blockchaina podijeljeni su sa svim čvorovima u sustavu i svaki čvor može vidjeti transakcije koje su provedene unutar blokova. Jednom kada se nešto zapiše u blockchain to se više ne može mijenjati, odnosno jako se teško može promijeniti, čime dobivamo nepromjenjiv zapis svih provedenih transakcija. Sustav je decentraliziran na veliku količinu pojedinih čvorova od kojih svatko sadrži najnovije podatke, što sam blockchain čini visoko pristupačnim. Svaka transakcija na blockchainu je kriptirana što mu daje sigurnost i integritet. Unutar financijske industrije blockchain može odigrati značajnu ulogu, omogućujući brže nagodbe i trgovinu, pošto ne podlaže dugotrajnim procesima verifikacije ispravnosti, raznim odobrenjima i izmirenjima od obiju stranaka te nije potrebno plaćati niti jednu treću stranku kako bi se zadržalo povjerenje tijekom transakcije. (Bashir, 2017.)

4.3.1. Ograničenja blockchaina

Kao i svaka tehnologija niti blockchain nije savršen i ima i svojih mana i ograničenja. Samo neki od njih odnose se na skalabilnost, adaptivnost, privatnost, način regulacije cijele mreže te sama činjenica da je blockchain relativno nova tehnologija i time još neistražena u potpunosti. (Bashir, 2017.)

Sve većom uporabom blockchaina i konstantnim povećanjem korisnika javlja se problem. Transakcije se gomilaju, a sistem ih ne može dovoljno brzo provesti. Sa svakom transakcijom u blockchain se dodaje novi blok, a svaki blok ima sve više i više podataka u sebi pošto sadrži povijest podataka blokova prije sebe.

Blockchain tehnologija trenutno je premlada kako bi se povezala i integrirala s postojećim sistemima, isto tako niti dva različita blockchaina ne mogu na jednostavan način međusobno komunicirati. Kako bi se tehnologija razvila i bila svakodnevno upotrebljiva na globalnom tržištu, trebao bi se razviti neki oblik standardizacije. Standardizacija omogućila bi puno bolju interoperabilnost i adaptivnost.

Transakcije koje se provode na blockchainu globalno su objavljene i nisu kriptirane u većini aplikacija koje se vrte na blockchainu. Ako se kod podataka vidljivih kod transakcija radi o osobnim podacima, primjerice zdravstvenim ili financijskim podacima, javlja se problem koji ovisno o državi može biti i legalne prirode. Tome bi se moglo stati na kraj da se kroz transakcije šalju samo kriptirani podaci, međutim onda se javlja problem što ako se izgubi ključ za dekripciju određenih podataka, isto tako ako je dekripcijski ključ ukraden i podaci su objavljeni, svi podaci će zauvijek biti dekriptirano objavljeni na blockchainu. (Bashir, 2017., dotmagazine.online: Security and privacy in blockchain environments, 2018.)

Ovisno o dijelu svijeta blockchain je drugačije reguliran i to predstavlja veliki problem što se tiče globalizacije i globalnog tržišta poslovanja. Primjerice, u Europskoj Uniji javlja se problem s javnim blockchain platformama i novoj „General Data Proetection Regulation (GDPR)“ regulativi o zaštiti osobnih podataka. Pošto podaci koji se objavljuju na javnom blockchainu su svima dostupni, te ako bi se radilo o nekim osobnim podacima prema novoj regulativi o zaštiti podataka, ne bi ih se smjelo otkrivati. U istočnoj Aziiji vrijedi tako zvani „bussines first regulation later“ pristup prema kojem je blockchain kompanijama odobreno poslovanje bez ikakvih restrikcija. Kina s druge strane ima potpunu restrikciju poslovanja s kriptovalutama, na taj način ograničivši i samu blockchain tehnologiju i njen napredak. Dok zemlje poput Južne Koreje prihvaćaju blockchain tehnologiju i njene moguće inovacije međutim poslovanje koristeći iste i dalje nije nikako regulirano.
(Bashir I, 2017.; fortune.com: How Should We Regulate Blockchain, 2018.)

Zaključimo za kraj da je blockchain jedan jako zanimljivi koncept kojem bi se moglo pojednostaviti i ne obavezno i olakšati poslovanje na globalnoj razini ukoliko se uspije doći do sporazuma oko njegove regulacije i riješi problem skalabilnosti koji u trenutku guši veliku većinu prometa na blockchainu.

5. Pametni ugovori

Pametni ugovori kao koncept pojavili su se još 1990-ih godina kada ih je teoritizirao Nick Szabo, međutim trebalo je dodatnih dvadesetak godina kako bi se shvatila njihova vrijednost i korist. Prema Szabo-*u* pametan ugovor je računalni transakcijski protokol koji ispunjava uvjete ugovora. Glavni cilj je zadovoljiti uobičajene uvijete klasičnih ugovora, kao što su na primjer uvjeti plaćanja, pravo zaloga, povjerljivost pa čak i ovrhe. Na taj način smanjiti mogućnosti odstupanja od ugovora, smanjiti potrebu za trećom strankom kao pouzdanog posrednika, smanjiti mogućnost prevare i transakcijske troškove. (Bashir I., 2017.; Szabo N., 1996.)

Pametan ugovor je računalni kod koji predstavlja dogovor koji se automatski izvršava ukoliko su ostvareni određeni uvjeti. Provedivost u smislu pametnog ugovora ne podrazumijeva samo klasičnu provedivost kao kod pisanih ugovora, u smislu da podlažu zakonu i da se njihovom provedbom osiguravaju određeni uvjeti i restrikcije, već bi trebali funkcionirati po principu ono što je napisano je zakon, odnosno kod je zakon. Čime bi se eliminirala potreba za trećom stranom kao svjedokom pri provođenju pametnog ugovora. Pametni ugovori su samoprovodni i sigurni, što bi značilo da bi trebali biti konstruirani s tolerancijom na greške i provedivosti u razumljivom vremenu. Pod toleranciju na greške smatra se da ako bi se pametan ugovor izvršio u nekoj okolini koja u sebi ima neke greške ili u kojoj se određeni programi ponašaju drugačije od očekivanog, to ne bi smjelo utjecati na pametni ugovor i njegovo izvršavanje bi trebalo biti konačno i ne izmijenjeno. Pametni ugovori rade koristeći konačni automat za regulaciju svojeg koda, što omogućuje izradu efektivnog programskog okruženja (eng. Framework) za programiranje pametnih ugovora, gdje se stanje ugovora nadograđuje ovisno o predefiniranim kriterijima i uvjetima. Drugim riječima pametan ugovor trebao bi zadovoljavati sljedeća svojstva:

- Automatska egzekucija
- Provedivost
- Interoperabilnost i otpornost na vanjske greške u sustavu
- Sigurni i nezaustavljivi nakon što su jednom pokrenuti

(Bashir I., 2017.; blockgeeks.com: Smart Contracts: The Blockchain Technology That Will Replace Lawyers, 2018.)

U smislu blockchaina pametan ugovor najjednostavnije rečeno čini program koji se izvršava na blockchainu koji u sebi ima implementiranu neku poslovnu logiku.

Mogu uključivati operacije poput ažuriranja podataka u najjednostavnijem smislu ili izvršavati određene operacije koje ovise o raznim u naprijed definiranim uvjetima. Na primjer kompleksniji pametan ugovor može utvrditi da li cijena transporta određenog proizvoda ovisi i koliko ovisi o tome gdje se proizvod transportira i u koje vrijeme stiže na odredište. (hyperledger.org: Hyperledger Architecture, Volume II,2018.)

Trenutno razlikujemo dva tipa pametnih ugovora:

1. Instalirani pametni ugovori - poslovna logika dodana je u mrežu prije nego je mreža pokrenuta.
2. Pametni ugovori koji se izvršavaju na mreži – izvršavanje poslovne logike tijekom provođenja određene transakcije.

Pametni ugovori obrađuju se na način da svaki pametni ugovor ima svoj jedinstveni identifikator, koji se zatraži prilikom provođenja neke transakcije, a u obzir se uzimaju i trenutno stanje blockchaina i ostale zavisnosti koje su postavljene na mreži. Određeni rezultat je generiran ukoliko je zahtjev bio validan i prihvaćen. Generiranje rezultata stvara novo stanje blockchaina koje se onda uzima u obzir u daljnjim operacijama s mrežom. (hyperledger.org: Hyperledger Architecture, Volume II,2018.)

Pošto se u daljnjem tekstu obrađuje Hyperledger i primjeri će biti prikazani koristeći Hyperledger Fabric pojasnimo oblik pametnih ugovora koji se provode na Hyperledger Fabric blockchainu.

5.1. Pametni ugovori na platformi Hyperledger Fabric (Chaincode)

Pametni ugovori za Hyperledger fabric mogu biti pisani koristeći Go programski jezik ili Nodejs. Chaincode se izvršava u sigurnom okruženju Docker kontejnera. Chaincode inicijalizira i upravlja stanjem blockchaina kroz transakcije koje provode određene aplikacije. Stanje koje se postigne izvršavanjem određenog Chaincode-a odnosi se isključivo na taj Chaincode i nije dostupno nekom drugom Chaincode-u, međutim uz određeno dopuštenje unutar mreže, unutar jednog Chaincode-a može se pozvati neki drugi Chaincode.

Razlikujemo Sistemski i Aplikacijski Chaincode:

Sistemski Chaincode barata sistemskim transakcijama kao što su upravljanje životnim ciklusom transakcije i definiranje uvjeta pod kojima se transakcija izvršava.

Aplikacijski Chaincode upravlja stanjem aplikacije na blockchainu, na primjer podacima koji se zapisuju u blok kroz neku aplikaciju.

Chaincode započinje s paketom koji sadrži metapodatke kao što su ime, verzija i digitalni potpisi druge strane kako bi se osigurao integritet koda i metapodataka. Chaincode paket se zatim instalira na čvorove u mreži koji vrše transakciju.

Član mreže zatim aktivira Chaincode na način da započne izvršavanje transakcije na mreži. Ako je transakcija odobrena, Chaincode ulazi u aktivno stanje u kojem onda može zaprimiti transakciju od korisnika, preko klijentske aplikacije. Svaka transakcija koja uključuje Chaincode se nadodaje na blockkchain i na taj način mijenja stanje blockchaina. (hyperledger.org: Hyperledger Architecture, Volume II,2018.)

5.2. Pokazivači

Nakon upoznavanja s pametnim ugovorima, postavlja se pitanje kako bi pametni ugovori pristupili podacima unutar nekog sustava koji bi im bili potrebi za provođenje određenih uvjeta zadanih pametnim ugovorom. Kao rješenje tom problemu pojavili su se tako zvani „pokazivači“ (eng. Oracle). Pokazivači osiguravaju podatke kojima se koristi pametni ugovor. Mogli bi ih nazvati „očima“ pametnih ugovora pošto oni određuju što pametni ugovor vidi odnosno s kojim podacima raspolaže, također određuje što pametan ugovor može izvršiti s dostupnim podacima. Pokazivači također mogu digitalno potpisivati podatke ukoliko je izvor podataka autentičan. Tada se pametan ugovor može „pretplatiti“ na pokazivač i povlačiti (eng. pull) podatke s njega ili pokazivač može opskrbljivati pametan ugovor s podacima (eng. push). Također pokazivači ne smiju biti sposobni manipulirati podacima koje sadrže već ih samo dostavljati pametnom ugovoru.

Međutim opisani oblik pokazivača nije decentraliziran već podatke vuče s nekog centralnog servera, te se opet pojavljuje problem povjerenja. Kako vjerovati nekom poduzeću da pametne ugovore opskrbljuje s ispravnim podacima ? Kako bi se to izbjeglo potrebno je i pokazivače decentralizirati te se javlja novi pojam, decentralizirani pokazivač. Decentralizirani pokazivač se bazira na povlačenju podataka s nekog drugog blockchaina ili distribuiranog sustava kako bi podaci ostali autentični. Iako dobro zamišljeno ideja s decentraliziranim pokazivačima trenutno je teško provediva pošto sami blockchainovi nisu standardizirani i komunikacija između dva različita blockchaina je teško ostvariva, osim ako bi se napravio neki privatni blcockchain s kojeg bi pokazivač vukao podatke što je i dalje na neki način centralizirano rješenje. (Bashir I., 2017.; Medium.com: The Oracle Problem, 2018.; Slobodnik J, 2018.)

6. Distribuirane i decentralizirane aplikacije

Distribuirane aplikacije, oblik su aplikacija koje se vrte na više računala istovremeno unutar neke mreže. Za razliku od u običajnih aplikacija koje se vrte na jednom računalu, distribuirane aplikacije vrte se na više računala istovremeno kako bi izvršile neki zadatak. Distribuirane aplikacije mogu komunicirati s više servera ili računala unutar iste mreže od bilo kuda na svijetu, a podijeljene su u dva dijela: klijentski i serverski. Pogodnost je u tome da ako se računalo ili server s kojeg se vrti distribuirana aplikacija sruši aplikacija se jednostavno pokrene s nekog drugog računala u mreži. Također distribuirane aplikacije omogućuju da više korisnika istovremeno koriste istu aplikaciju.
Decentralizirane aplikacije, skupni je naziv za oblik distribuiranih aplikacija koje se vrte na blockchainu. Programirane su uz pomoć pametnih ugovora odnosno programskog koda smještenog na blockchainu koji predstavlja neku poslovnu logiku koju one onda izvode. (Techtarget.com: Distributed applications, 2018.)

Distribuirane aplikacije mogu biti izvedene koristeći četiri tipa mrežne arhitekture, koje uključuju:

- Dvoslojnu arhitekturu
- Troslojnu arhitekturu
- Uslužno orijentiranu arhitekturu
- Peer to peer arhitekturu (P2P)

Dvoslojna arhitektura, oblik je softverske arhitekture u kojoj se prezentacijski sloj vrti na klijentu, a podatkovni sloj se vrti na nekom udaljenom serveru, klijent i server međusobno komuniciraju preko nekog zahtjev/odgovor protokola. Vidljivo je da u dvoslojnoj arhitekturi nema dijela koji bi se bavio poslovnim slojem na kojem bi bila implementirana poslovna logika sustava, odnosno kod ovakve arhitekture poslovna logika ravnopravno je podijeljena između klijenta i servera. Dobra strana ovakve implementacije uglavnom je jednostavnost implementacije. (techopedia.com: Two-Tier Architecture, 2018.)

Kao što i samo ime kaže troslojna arhitektura tip je softverske arhitekture koja se sastoji od tri sloja, iako se i dalje radi o klijent/server arhitekturi poslovna logika odvojena je u posebnom aplikacijskom sloju, dok ostala dva sloja čine prezentacijski sloj koji je vidljiv korisniku i podatkovni sloj koji se vrti na nekom serveru. Velika prednost kod ove arhitekture u odnosu na dvoslojnu arhitekturu je ta da se svaki sloj može mijenjati zasebno bez da utječe na ostala dva sloja. Tako možemo odlučiti osvježiti web mjesto s novim izgledom bez da mijenjamo logiku ili podatke upisane na serveru. (techopedia.com: Three-Tier Architecture, 2018.)

Uslužno orijentiranu arhitekturu, čini skupina usluga koje međusobno komuniciraju. Komunikacija može uključivati jednostavnu podjelu podataka ili može uključivati izvršavanje neke radnje kroz koordinaciju dvije ili više usluga. Usluga čini dobro definirana, samoodrživa funkcionalnost koja ne ovisi o kontekstu ili stanju ostalih usluga. Usluge se pružaju kroz neki komunikacijski protokol preko mreže. Prema uslužno orijentiranoj arhitekturi usluga ima četiri definirana svojstva: logički reprezentira poslovnu aktivnost s određenim ishodom, samoodrživa je, kupac je vidi kao crnu kutiju i može se sastojati od više usluga. Uslužno orijentirana aplikacija sastoji se od više usluga uglavnom hijerarhijski raspoređene.
(Ibm.com: Service-oriented Architecture, 2018; opengroup.org: Service-Oriented Architecture Standards,2018., opengropu.org: What is SOA, 2018.)

U ovom radu najzanimljivija je Peer to peer arhitektura (P2P), pošto će se i kasnije navedeni primjeri u tekstu odnositi i provoditi na Peer to peer arhitekturi, pa da se ukratko pojasni.
Radi se o često korištenom tipu mrežne arhitekture u kojem svaki čvor, sudionik mreže ima iste sposobnosti i odgovornosti. Ime također može označavati i jedan program (aplikaciju) dizajniran na način da svaka instanca programa istovremeno reprezentira i server i klijenta. Često se koristi za podjelu sadržaja, multimediju i naravno blockchain i aplikacije koje se vrte na njemu također su zasnovane na Peer to peer arhitekturi. Zašto je Peer to peer efikasan, ako želimo nešto preuzeti koristeći P2P mrežnu infrastrukturu klijent pristupa podacima s više čvorova u mreži i od svakog čvora preuzima dio podataka. Za razliku od klasičnog servera koji je limitiran određenom propusnošću preuzimanja, više čvorova istovremeno povećava brzinu preuzimanja na način da se dio podataka preuzima s više izvora istovremeno. Zaključujemo da se povećanjem mreže povećava i efikasnost preuzimanja. Tolerancija na kvarove i maliciozne čvorove još jedna je važna karakteristika P2P mrežne arhitekture, najme s pojavom kvara na nekom čvoru unutar mreže čvor se jednostavno izostavlja i podaci se preuzimaju od drugih čvorova, eliminira se tako zvano usko grlo sustava. (techopedia.com: Peer-to-Peer Arhitecture, 2018.)

7. Hyperledger

Današnje poslovanje ovisi o kompleksnim i velikim bazama podataka koje ne rijetko moraju biti dijeljene unutar organizacije odnosno čijim podacima moraju imati pristup svi članovi organizacije istovremeno u bilo koje doba, kako bi mogli obavljati zadatke ili kako bi bili upućeni na daljnje promjene u poslovanju. Kako bi se tome doskočilo stvorene su distribuirane baze podataka koje su omogućile da više korisnika istovremeno pristupaju istom podatku. Međutim dijeljene baze podataka nameću pitanja:

Kome vjerovati oko podjele podataka ?
Kako biti siguran da su podijeljeni podaci ispravni ?
Kako biti siguran da je osoba na mreži i ona osoba za koju se predstavlja ?
Tko smije i na koji način upravljati i mijenjati bazu podataka ?
Što će se dogoditi ako dvoje ljudi unutar organizacije žele istovremeno promijeniti isti podatak?

Kroz godine ljudi su doskakali raznim dosjetkama kako bi riješili navedena pitanja, a jedna nova ideja koja se nedavno pojavila kako bi odgovorili na zadana pitanja upravo je blockchain tehnologija.

Jedan od oblika blockchaina koji pruža odgovore na navedena pitanja je i Hyperledger, osnovan 2015. godine. Hyperledger je open source pokušaj kreiranja napredne blockchain tehnologije za povezivanje više različitih industrija. Radi se o suradnji između industrija kao što su bankarstvo, financije i proizvodnih kompanija koje se bave primjerice pametnim uređajima i IOT-om (Internet of things), a razvija se pod pokroviteljstvom Linux zaklade.

Blockchain može imati razne zahtjeve za razne slučajeve upotrebe, primjerice ako sudionici imaju već uspostavljeno međusobno povjerenje kao na primjer dvije financijske institucije koji međusobno već imaju potpisane zakonski sporazum, blockchain se može prilagoditi na način da se provodi blaži algoritam konsenzusa između sudionika kako bi se ubrzala transakcija, dok s druge strane ako je povjerenje među strankama minimalno transakcija bi se odužila zbog dodavanja dodanog stupnja sigurnosti.

Hyperledger uzima u obzir puni spektar mogućnosti u poslovanju i razumije da razne organizacije imaju različite potrebe kojima se treba prilagoditi. Kako bi se prilagodili Hyperledger projekti slijede određenu filozofiju. Svi njihovi proizvodi moraju biti: Prilagodljivi, visoko osigurani, interoperabilni, neovisni o kriptovalutama i imati svoje API-je koji bi bili dostupni javnosti. (hyperledger.org: An Introduction to Hyperledger, 2018.)

Hyperledger stvara prilagodljive, nadogradive razvojne okvire (eng. framework) s općim razvojnim blokovima koji su ponovo upotrebljivi. Takav oblik prilagodljivosti omogućuje razvojnim inženjerima eksperimentiranje s raznim sastavnim dijelovima i mijenjanje određenih dijelova bez da utječu na ostatak sustava. Što omogućava prilagođavanje sustava javnih knjiga na razne uvijete. Isto tako više razvojnih inženjera može istovremeno raditi na različitim modulima i ponovo koristiti opće razvojne blokove kroz više projekata.

Kao što je već uspostavljeno ranije sigurnost je ključna unutar bilo kojeg oblika blockchaina ili javne knjige, pošto većina transakcija uključuje vrijedne transakcije sa osjetljivim podacima. Sigurnost i robustnost ključni su kako bi se osigurao razvoj među korporacijskim blockchainova.

Interoperabilnost, već je spomenuta kao problem koji se javlja unutar blockchaina, kojeg je uočio i Hyperledger i želi mu stati na kraj. Treba uzeti u obzir da ako se blockchain tehnologija kao takva probije da će veliki broj različitih blockchainova morati međusobno komunicirati i razmjenjivati podatke, zbog čega se Hyperledger zalaže za mogućnost prenosivosti svojih pametnih ugovora na druge blockchainove.

Neovisnost o kripto valuti, odnosi se na mogućnost upotrebljivosti blockchaina i bez postojanja i uporabe određene kriptovalute, iako i dalje pruža mogućnost stvaranja kriptovaluti ukoliko to korisnik želi. Za kraj Hyperledger pruža API-je koji omogućuju interoperabilnost između ostalih sustava.
(hyperledger.org: An Introduction to Hyperledger, 2018.)

7.1. Uporaba Hyperledger blockchaina u praktične svrhe

Ovo poglavlje opisati će kroz par primjera upotrebu blockchaina kao rješenja za svakidašnjih problema, problemi se javljaju iz raznih izvora i industrija koje nas svakodnevno okružuju i s kojima smo u dodiru.

- Bankarstvo – podizanje kredita
- Financije – procesi nakon izvršavanja transakcije
- Zdravstvo – vjerodostojnost liječnika

Podizanje kredita

Kada banke izdaju kredite žele se osigurati da će izdani krediti biti i vraćeni te svakog tko dođe tražiti kredit temeljito analiziraju kako bi ustanovili da li osoba odgovara traženim uvjetima, pri tome o osobi sakupljaju veliku količinu osobnih podataka, što banke čini čestom metom hakera, koji kroz osobne podatke analiziraju i biraju svoje žrtve. Još većem riziku se ljudi izlažu ako kredite traže u različitim bankama, jer na taj način svoje informacije distribuiraju još većem krugu ljudi. Hyperledger nudi upravo rješenje za to. Hyperledger omogućuje da banke dijele samo određene informacije koje bi banki omogućile donesti pravu odluku o posuđivanju novca, a sve dobivene informacije bile bi kriptirane i distribuirane, što bi uvelike otežalo neovlašteno povlačenje informacija s mreže.

Post transakcijski procesi

U današnje vrijeme post transakcijski procesi uključuju veći broj procesa koji se uglavnom odrađuju po nekom protokolu koji je podijeljen kroz više odjela neke organizacije na više ljudi od kojih svako odrađuje određeni dio validacije transakcije.

Blockchain rješenje nudi efikasniji put do istog cilja. Jedna strana u čvor ubaci podatke koje druga strana verificira, a sama mreža djeluje kao treća strana koja osigurava povjerenje da ulazni podaci nisu izmijenjeni na bilo koji način. Nadalje svi podaci se mogu spremiti na blockchain tako da su u svakom trenutku dostupni obojim stranama.

Vjerodostojnost liječnika

Blockchain nastoji stati na kraj i raznim dugotrajnim procesima verifikacije i provjera u određenim zanimanjima, primjerice kod zapošljavanja liječnika.

Kada se liječnik odluči zaposliti u novoj bolnici on sa sobom ostavlja dugi papirni trag. Od diplome, certifikata, specijalizacija, dozvole za vršenje liječničke prakse, preporuka i razloga zašto mijenja službu do detalja o kažnjavanju ili zlouporabi službene dužnosti. S druge strane svaka bolnica prije nego zaposli novog liječnika primorana je na dubinsku istragu svojih budućih zaposlenika. Jednom kada je sva dokumentacija prihvaćena i odobrena bolničko

povjerenstvo za vjerodostojnost odlučuje da li osoba može započeti raditi na novom radnom mjestu. Cijeli taj postupak je kompleksan, dugotrajan i manjka povjerenja.
Blockchain u ovom slučaju nudi rješenje na sljedeći način:

1. Liječnik traži dokaz o završenom studiju od strane fakulteta
2. Nakon povezivanja fakulteta i liječnika unutar blockchain, fakultet liječniku šalje uvjerenje kao dokaz o završenom fakultetu i ostalim potrebnim informacijama. Taj dokaz je spremljen u liječnikovom krajnjem čvoru (računalu) koji je spojen na blockchain, ali niti jedan dio podataka nije izravno spremljen na blockchain. Blockchain sadrži samo dokaze u obliku javnih identifikatora da liječnik posjeduje određene dokumente.
3. Liječnik zatim može dijeliti određene podatke sa svojih uvjerenja, na primjer potvrdu o završenom fakultetu i učiniti je javno dostupnom bolnici. Pošto je sve unaprijed potpisano od strane fakulteta bolnica može preko blockchaina izvršiti verifikaciju izvora.

Ovim oblikom verifikacije štite se privatni podaci liječnika a istovremeno se traženoj organizaciji pružaju svi potrebni podaci, pritom se ubrzava proces verifikacije i smanjuje papirnati trag. (hyperledger.org: An Introduction to Hyperledger, 2018.)

Iako nudi veliki broj rješenja treba shvatiti da je nelogično, možda i ne moguće da samo jedan blockchain može doskočiti svim tim zadacima, zbog čega Hyperledger razvija više oblika radnih okruženja i alata koji se izvršavaju na njihovom blockchainu od kojih je svaki prilagođen i specijaliziran za određena područja. U nastavku slijedi kratki prikaz raznih radnih okruženja koje Hyperledger podržava.

7.2. Hyperledger Burrow

Hyperledger Burrow pruža modalni blockchain klijent sa interpreterom pametnih ugovora razvijen prema Ethereumovom virtualnom stroju. Što znači da omogućuje provođenje bilo kojeg oblika pametnih ugovora koji su se ili se mogu provoditi na Ethereum virtualnom stroju.

Sastoji se od sljedećih stavki:

- Konsenzusni mehanizam – održava mrežni stog između čvorova i određuje koje će se transakcije provesti kroz aplikacijski mehanizam.
- Aplikacijsko blockchain sučelje – pruža sučelje preko kojeg se povezuju i međusobno komuniciraju konsenzusni i aplikacijski mehanizam.
- Mehanizam za pametne ugovore – pruža razvoj kompleksnih pametnih ugovora za provođenje kompleksnih poslovnih procesa.
- Mrežni pristupnik – koji pruža programabilno sučelje za komunikaciju s mrežom i korisničkim sučeljem.

7.3. Hyperledger Fabric

Hyperledger fabric je platforma za izgradnju blockchain rješenja s modularnom arhitekturom koja pruža visoki stupanj povjerljivosti, fleksibilnosti i skalabilnosti, što ga čini prihvatljivim za razne grane industrije.

Upotrebljava kontejnere u kojima provodi pametne ugovore koji se u slučaju Fabrica zovu „lančani kod“ (eng. Chaincode) koji sadrže poslovna pravila sustava. Podržava više oblika konsenzusa što ga čini prilagodljivim za više slučajeva upotrebe i razine povjerenja. Ne oslanja se na niti jedan oblik kriptovaluta i omogućava pisanje pametnih ugovora u već poznatim i korištenim programskim jezicima kao što su: Javascript, Go, Python i Java. Nadalje Fabric koristi pojam članstva koji je prenosiv, pošto podržava procese autorizacije i lanac zapovijedanja. Omogućuje stvaranje privatnih kanala na blockchainu na kojem sudionici mogu provoditi transakcije bez da utječu na globalni blockchain.

7.4. Hyperledger Indy

Hyperledger Indy čini blockchain izrađen za upravljanje identitetom. Omogućuje stvaranje i uporabu neovisnih digitalnih identiteta koji se nalaze na blockchainu.

Indy odgovara na pitanja:

S kim se posluje ?
Kako verificirati podatke koje druga strana daje na raspolaganje ?

Glavne karakteristike Hyperledger Indy-a:

- Suverenitet nad vlastitim podacima – Indy sprema podatke o identitetu na javni blockchainu, podaci mogu uključivati javne ključeve, dokaze o postojanju i slično, međutim samo vlasnik podataka podatke može mijenjati ili ih maknuti.
- Privatnost - Privatnost je osigurana na način da svaki korisnik ostavlja samo točno određene podatke koje on smatra da smiju biti eksponirani bez ikakve korelacije s ostalim podacima korisnika.
- Provjerljivi zahtjevi – Indy bi omogućio uporabu točno određenih podataka za identifikaciju ovisno o kontekstu i podacima koje kontekst zahtjeva.

7.5. Hyperledger Iroha

Hyperledger Iroha tip je blockchain radnog okruženja, koji je se jednostavno integrira u neku infrastrukturu koja zahtjeva postojanje blockchaina.
Glavne karakteristike:

- Jednostavna struktura
- Moderan i dizajn koji u pozadini vrti C++
- Glavni naglasak na mobilnu tehnologiju

7.6. Hyperledger Sawtooth

Hyperledger Sawtooth je modalna platforma za izgradnju, razvoj i provođenje blockchaina. Novonastali blockchain sadrži digitalni zapis koji se održava bez centralnog autoriteta. Glavna zamisao je da blockchain ostane distribuiran i decentraliziran, a da pametni ugovori budu sigurni za poduzetno korištenje.

Glavne karakteristike:

- Dinamički konsenzus - omogućuje promjenu konsenzusna na blockchainu provodom određene transakcije.
- Konsenzus potrošenog vremena (eng. Proof of elapsed time) – Konsenzusni algoritam sa skalabilnošću Proof of Work algoritma bez trošenja tolike količine električne energije.
- Transakcijske obitelji – Omogućuje korisnicima pisanje pametnih ugovora u bilo kojem željenom programskom jeziku
- Paralelno izvođenje transakcija - Omogućuje podjelu blokova u paralelne tokove
- Privatne transakcije – Grupa Sawooth čvorova može se izvršiti sa različitim dozvolama i dopuštenjima.

(hyperledger.org: An Introduction to Hyperledger, 2018.)

8. Prikaz izvršavanja decentralizirane aplikacije u Hyperledger Fabric-u (na linux operacijskom sustavu)

8.1. Lokalni blockchain

Započnimo prvo sa stvaranjem vlastite mreže odnosno, podignimo lokalni Hyperledger fabric blockchain.

Prije nego se započne potrebno je instalirati:

- Docker (moguće je preuzeti s https://docs.docker.com/install/)
- Go programski jezik (moguće je preuzeti s https://golang.org/doc/install)

Prvo kloniramo službene Hyperledger fabrics uzorke, pri tome pazeći na traženu verziju, za vrijeme pisanja ovog rada zadnja stabilna verzija bila je Hyperledger 1.2.0. Preuzeti sve potrebne binarne datoteke možemo s naredbom:

```
curl -sSL http://bit.ly/2ysbOFE | bash -s 1.2.0
```

ukoliko je dođe do pogreške provjerite da li imate zadnju verziju curl-a.

Nakon uspješnog preuzimanja uzoraka prebacimo se u direktorij first-network. Koristimo naredbu

```
cd fabric-samples/first-network
```

Sljedeće postavimo okruženje sa putanjom do binarnih datoteka kako bi Fabric znao gdje ih pronaći . To možemo preko sljedeće naredbe.

```
export PATH=<zamijeniti sa svojom putanjom>/bin:$PATH
```

Potpunu putanju možemo vidjeti tako da unutar direktorija provedemo naredbu pwd. Sada možemo krenuti na postavljanje lokalnog blockchaina. Moramo kreirati sudionike u mreži. Svaki sudionik mora imati jedinstveni i sigurni identifikator.

```
../bin/cryptogen generate --config=./crypto-config.yaml
```

Provedbom gornje naredbe pokrećemo cryptogen alat, preko kojeg pokrećemo konfiguracijsku datoteku crypto-config.yaml koja generira dvije organizacije.

```
org1.example.com
org2.example.com
```

Sljedeće pokrećemo

```
export FABRIC_CFG_PATH=$PWD

../bin/configtxgen -profile TwoOrgsOrdererGenesis -outputBlock ./channel-artifacts/genesis.block
```

Trebalo bi se prikazati sljedeće

```
 2018-08-17 14:04:13.910 CEST [common/tools/configtxgen] main -> INFO 002 Loading configuration
2018-08-17 14:04:13.933 CEST [msp] getMspConfig -> INFO 006 Loading NodeOUs
2018-08-17 14:04:13.934 CEST [msp] getMspConfig -> INFO 008 Loading NodeOUs
2018-08-17 14:04:13.934 CEST [common/tools/configtxgen] doOutputBlock -> INFO 00a Generating genesis block
2018-08-17 14:04:13.934 CEST [common/tools/configtxgen] doOutputBlock -> INFO 00b Writing genesis block
```

Pri ovom koraku stvorili smo dvije organizacije i stvorili smo dva čvora po organizaciji.

Stvorili smo certifikate za obije organizacije, što znači da se svaka transakcija može potpisati od strane organizacije.

Na taj način znamo tko je kreirao i potpisao transakciju.

Kreira se tako zvani Genesis blok, odnosno prvi blok u lancu.

Sljedeće treba kreirati kanal unutar kojeg organizacije mogu međusobno komunicirati. U ovom slučaju nazvat ćemo ga kanal1, slobodno ga nazovite po želji.

```
export CHANNEL_NAME=kanal1  && ../bin/configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx -channelID $CHANNEL_NAME

../bin/configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID $CHANNEL_NAME -asOrg Org1MSP

../bin/configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID $CHANNEL_NAME -asOrg Org2MSP
```

Zadnje dvije provedene naredbe, stvaraju sidrena čvorišta kako bi novi pridošlice u mrežu mogli komunicirati s njom i doznati koji su ostali sudionici u mreži.

Krenimo napokon u podizanje mreže. Za podizanje mreže koristit ćemo docker.

```
docker-compose -f docker-compose-cli.yaml up -d
```

Dobijemo nešto slično ovome:

```
Creating network "net_byfn" with the default driver
Creating volume "net_orderer.example.com" with default driver
Creating volume "net_peer0.org1.example.com" with default driver
Creating volume "net_peer1.org1.example.com" with default driver
Creating volume "net_peer0.org2.example.com" with default driver
Creating volume "net_peer1.org2.example.com" with default driver
Creating peer1.org1.example.com ... done
Creating peer0.org1.example.com ... done
Creating peer1.org2.example.com ... done
Creating orderer.example.com    ... done
Creating peer0.org2.example.com ... done
Creating cli                    ... done  (cli – comand line interface)
```

Dalje postavljamo Docker okruženje kako bi u njemu mogli zadavati i provoditi transakcije

```
docker start cli
```

Uđemo u docker kontejner

```
docker exec -it cli bash
```

Kada se pojavi nešto slično ušli smo u kontejner.

```
root@81af69a1126e:/opt/gopath/src/github.com/Hyperledger/fabric/peer#
```

Sada u kontejner dodajemo konfiguracije kanala koje smo maloprije postavili, kako bi kontejner mogao započeti s izvršavanjem kanala. Eksportamo ime kanala

```
export CHANNEL_NAME=kanal1
```

i kreiramo novi kanal unutar kontejnera sa zadanim konfiguracijama.

```
peer channel create -o orderer.example.com:7050 -c $CHANNEL_NAME -f
./channel-artifacts/channel.tx --tls --cafile
/opt/gopath/src/github.com/Hyperledger/fabric/peer/crypto/ordererOrganizati
ons/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.c
om-cert.pem
```

Zatim dodajemo čvorove u kanal.

```
peer channel join -b mychannel.block
```

Provođenjem gornje naredbe pridružujemo organizaciju prvom čvoru u mreži.

Drugu organizaciju pridružujemo istom čvoru.

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/Hyperledger/fabric/peer/
crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer0.org2.example.com:7051
CORE_PEER_LOCALMSPID="Org2MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/Hyperledger/fabric/p
eer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/
tls/ca.crt peer channel join -b mychannel.block
```

Ažuriramo okolne varijable kako bi ih sustav prepoznao.

```
peer channel update -o orderer.example.com:7050 -c $CHANNEL_NAME -f
./channel-artifacts/Org1MSPanchors.tx --tls --cafile
/opt/gopath/src/github.com/Hyperledger/fabric/peer/crypto/ordererOrganizati
ons/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.c
om-cert.pem
```

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/Hyperledger/fabric/peer/
crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer0.org2.example.com:7051
CORE_PEER_LOCALMSPID="Org2MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/Hyperledger/fabric/p
eer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/
tls/ca.crt peer channel update -o orderer.example.com:7050 -c $CHANNEL_NAME
-f ./channel-artifacts/Org2MSPanchors.tx --tls --cafile
/opt/gopath/src/github.com/Hyperledger/fabric/peer/crypto/ordererOrganizati
ons/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.c
om-cert.pem
```

Čvorove smo postavili kao sidrene čvorove kako bi ostali čvorovi u mreži mogli s njima komunicirati.

Instalacija pametnih ugovora na lokalni blockchain

U Hyperledger Fabric-u pametni ugovori još se zovu i *chaincode*. Instalirati ćemo u naprijed priređene pametne ugovore koje možemo preuzeti s Hyperledger fabric git repozitorija.

```
peer chaincode install -n mycc -v 1.2.0 -p
github.com/chaincode/chaincode_example02/go/
```

```
peer chaincode instantiate -o orderer.example.com:7050 --tls --cafile
/opt/gopath/src/github.com/Hyperledger/fabric/peer/crypto/ordererOrganizati
ons/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.c
om-cert.pem -C $CHANNEL_NAME -n mycc -v 1.2.0 -c '{"Args":["init","a",
"100", "b","200"]}' -P "OR ('Org1MSP.peer','Org2MSP.peer')"
```

Izvršavanjem druge naredbe čvorovima smo preko pametnog ugovora osigurali tako zvane *„tokene“* s kojima se naknadno mogu koristiti u transakcijama između organizacija, čvoru a osigurali smo 100 tokena a čvoru b 200 tokena.

Upravo smo završili lokalni blockchain koji se trenutno sastoji od dva čvora koji mogu međusobno komunicirati i provoditi transakcije. Provjerimo još jednom eksplicitno stanje svakog čvora.

```
peer chaincode query -C $CHANNEL_NAME -n mycc -c '{"Args":["query","a"]}'
```

Pošaljimo sad 10 tokena s čvora a na čvor b

```
peer chaincode invoke -o orderer.example.com:7050  --tls --cafile
/opt/gopath/src/github.com/Hyperledger/fabric/peer/crypto/ordererOrganizati
ons/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.c
om-cert.pem  -C $CHANNEL_NAME -n mycc -c '{"Args":["invoke","a","b","10"]}'
```

Provjerimo da li je transakcija uspješna

```
peer chaincode query -C $CHANNEL_NAME -n mycc -c '{"Args":["query","a"]}
```

Uspješno je podignut lokalni blockckchain na kojem je moguće vršiti transakcije. Dalje slijedi izgradnja jednostavne aplikacije koja će omogućavati pisanje, ažuriranje i provođenje upita na lokalnom blockchainu.

8.2. Decentralizirana aplikacija

Aplikacija će zapisivati vlasništva automobila i svatko u mreži moći će vidjeti koji automobil pripada kome.

Prije nego se krene na pregled aplikacije treba ukinuti mrežu koju smo u ranijim koracima postavili, treba počistiti docker kontejnere i docker mrežu, a to sve možemo napraviti sa sljedećim instrukcijama.

```
cd fabric-samples/first-network
first-network$./byfn.sh-mdown
first-network$ docker rm -f $(docker ps -aq)
first-network$ docker network prune
```

byfn.sh je unaprijed pripremljena skripta sa strane Hyperledger tima koja omogućava jednostavnije podizanje mreže i blockchaina, pokretanjem skripte s naznakom „down“ rušimo postojeću mrežu.

Nakon odrađenih kućanskih poslova prebacujemo se u direktorij fabcar, koji se nalazi u kloniranom git repozitoriju koji je preuzet još u koracima postavljanja mreže. Pokrećemo skriptu

```
./startFabric.sh
```

Pokrenuta skripta podiže mrežu, sljedeći korak je instalacija nodejs-a i odgovarajućeg SDK-a (eng. Software Development Kit) za Hyperledger, pošto su klijentske aplikacije pisane u javascriptu.

Najjednostavnija instalacija može se napraviti preko zvaničnog node-ovog paket menadžera (npm).

```
npm install
```

Npm instalirao je javascript biblioteke fabric-client i fabric-ca-client u node_module. Slijedi kreacija administratora, kako bi se kreirao administrator pokreće se skripta enrollAdmin.js

```
node enrollAdmin.js
```

Pokrenuta skripta komunicira s Hyperledger Fabric CA-om (eng certificate authority) kako bi generirala certifikat za administratora i onda pomoću tih certifikata se prijavila u mrežu. Certifikati se spremaju lokalno, kako bi se mogli koristiti kada aplikacija komunicira s mrežom.

Ako je administrator uspješno registriran dobije se sljedeća poruka:

```
Store path:/home/neven/fabric-samples/fabcar/hfc-key-store

Successfully enrolled admin user "admin"
Assigned the admin user to the fabric client
::{"name":"admin","mspid":"Org1MSP","roles":null,"affiliation":"","enrollme
ntSecret":"","enrollment":{"signingIdentity":"23d59b3967dd60701ae4af7e674b6
e0cb8ee56ad006d9df9a64e2df0dc3cf422","identity":{"certificate":"-----BEGIN
```

```
CERTIFICATE-----
\nMIICAjCCAaigAwIBAgIUR/eJmfFhdQoh0lW7WnmXSU31dp8wCgYIKoZIzj0EAwIw\nczELMAk
GA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExFjAUBgNVBAcTDVNh\nbiBGcmFuY2lzY28x
GTAXBgNVBAoTEG9yZzEuZXhhbXBsZS5jb20xHDAaBgNVBAMT\nE2NhLm9yZzEuZXhhbXBsZS5jb
20wHhcNMTgwODE5MTU1MDAwWhcNMTkwODE5MTU1\nNTAwWjAhMQ8wDQYDVQQLEwZjbGllbnQxDj
AMBgNVBAMTBWFkbWluMFkwEwYHKoZI\nzj0CAQYIKoZIzj0DAQcDQgAESpROflSXuuNotGnZxpG
BgKJX6HzClcZD7MvS0uV0\nr9DyiDziZympfQtM/XHSzsMExiv9XVc2p6kZ+kwXSwXJfKNsMGow
DgYDVR0PAQH/\nBAQDAgeAMAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFPbO30fnPRel2DgMDJqlI
C4h\nxE6jMCsGA1UdIwQkMCKAIEI5qg3NdtruuLoM2nAYUdFFBNMarRst3dusalc2Xkl8\nMAoG
CCqGSM49BAMCA0gAMEUCIQC17ykNFOrb3ISTvYSptFUKXviVNNqpGfbFyRyx\n3ry3nQIgQWPcQ
CkLi9WrNb36JKuwLYGztpx+FH4+0LUWdU7Ga7A=\n-----END CERTIFICATE-----\n"}}}
```


Nakon uspješne prijave administratora prijavljujemo korisnika, pokrećemo skriptu registerUser.js, dobivamo sljedeću poruku.

```
Store path:/home/neven/fabric-samples/fabcar/hfc-key-store
Successfully loaded admin from persistence
Successfully registered user1 - secret:NFjOOPJvzsgp
Successfully enrolled member user "user1"
User1 was successfully registered and enrolled and is ready to interact
with the fabric network
```

Ova skripta koristi administratorske ovlasti koje smo kreirali u prijašnjem koraku kako bi zahtijevao kreiranje novog korisnika.

Pošto aplikacija sprema podatke na blockchain, osnovna mogućnost koju nudi je postavljanje upita kako bi mogli čitati zapise. Za čitanje zapisa pokrenemo skriptu query.js Skripta query.js koristi kreirane korisničke podatke za postavljanje upita nad mrežom. Rezultat su svi auti koji su zapisani na blockchainu.

```
node query.js
```

```
Store path:/home/neven/fabric-samples/fabcar/hfc-key-store
Successfully loaded user1 from persistence
Query has completed, checking results
Response is [{"Key":"CAR0",
"Record":{"colour":"blue","make":"Toyota","model":"Prius","owner":"Tomoko"}
},{"Key":"CAR1",
"Record":{"colour":"red","make":"Ford","model":"Mustang","owner":"Brad"}},{
"Key":"CAR2",
"Record":{"colour":"green","make":"Hyundai","model":"Tucson","owner":"Jin
Soo"}},{"Key":"CAR3",
"Record":{"colour":"yellow","make":"Volkswagen","model":"Passat","owner":"M
ax"}},{"Key":"CAR4",
"Record":{"colour":"black","make":"Tesla","model":"S","owner":"Adriana"}},{
"Key":"CAR5",
"Record":{"colour":"purple","make":"Peugeot","model":"205","owner":"Michel"
}},{"Key":"CAR6",
"Record":{"colour":"white","make":"Chery","model":"S22L","owner":"Aarav"}},
{"Key":"CAR7",
"Record":{"colour":"violet","make":"Fiat","model":"Punto","owner":"Pari"}},
{"Key":"CAR8",
"Record":{"colour":"indigo","make":"Tata","model":"Nano","owner":"Valeria"}
},{"Key":"CAR9",
"Record":{"colour":"brown","make":"Holden","model":"Barina","owner":"Shotar
o"}}]
```

Važno je napomenuti da će svaki čvor u mreži koji pokrene query.js skriptu vidjeti potpuno isti rezultat, iako niti jedan od čvorova zapravo ne posjeduje podatke.

Otvaranjem skripte query.js može se vidjeti da je konkretan upit koji se prosljeđuje na blockchain sljedeći:

```
const request = {
 chaincodeId: 'fabcar',
 fcn: 'queryAllCars',
 args: ['']
};
```

U koliko se malo poigramo i izmijenimo kod na način,

```
const request = {
 chaincodeId: 'fabcar',
 fcn: 'queryCar',
 args: ['CAR6']
};
```

spremimo i ponovo pokrenemo skriptu, upit će izbaciti

```
Store path:/home/neven/fabric-samples/fabcar/hfc-key-store
Successfully loaded user1 from persistence
Query has completed, checking results
Response is
{"colour":"white","make":"Chery","model":"S22L","owner":"Aarav"}
```

Sve radi, a ako nas zanima kako točno izgleda pametan ugovor koji to omogućuje, želimo ga recimo promijeniti. Možemo ga provjeriti

```
../chaincode/fabcar/go/fabcar.go
```

Pogledajmo funkciju koja se koristila za traženje automobila po ključu.

```
func (s *SmartContract) queryCar(APIstub shim.ChaincodeStubInterface, args
[1]string) sc.Response {

        if len(args) != 1 {
                return shim.Error("Incorrect number of arguments. Expecting
1")
        }

        carAsBytes, _ := APIstub.GetState(args[0])
        return shim.Success(carAsBytes)
}
```

Funkcija prima jedan argument u našem slučaju ključ i po ključu na zaslon ispisuje podatke o automobilu.

Promotrimo još jednu funkciju.

```
func (s *SmartContract) changeCarOwner(APIstub shim.ChaincodeStubInterface,
args []string) sc.Response {

        if len(args) != 2 {
                return shim.Error("Incorrect number of arguments. Expecting
2")
        }

        carAsBytes, _ := APIstub.GetState(args[0])
        car := Car{}

        json.Unmarshal(carAsBytes, &car)
        car.Owner = args[1]

        carAsBytes, _ = json.Marshal(car)
        APIstub.PutState(args[0], carAsBytes)

        return shim.Success(nil)
}
```

Pozivanjem funkcije omogućuje se promjena vlasništva automobila koja se bilježi na blockchainu.

Međutim želimo i pisati po blockchainu, a ne samo čitati podatke s njega ili mijenjati već postojeće podatke. Ukoliko želimo dodati novi automobil na blockchain, pokrećemo skriptu invoke.js koja nam omogućava promjenu stanja blockchaina.

Pokretanjem invoke.js skripte dobivamo sljedeće:

```
Store path:/home/neven/fabric-samples/fabcar/hfc-key-store
Successfully loaded user1 from persistence
Assigning transaction_id:
efeaff9616a1f34b11b01c21f49bde346e69632f16f2d8f3510f74faa9eeba6e
Transaction proposal was bad
Failed to send Proposal or receive valid response. Response null or status
is not 200. exiting...
Failed to invoke successfully :: Error: Failed to send Proposal or receive
valid response. Response null or status is not 200. exiting...
```

Primjećuje se greška pošto skripta očekuje da se modificira s odgovarajućim argumentima kako bi mogla izvršiti pisanje na blockchain.

U sljedeći dio skripte dodaju se podaci koji će biti zapisani na blockchain.

```
var request = {
                //targets: let default to the peer assigned to the client
                chaincodeId: 'fabcar',
                fcn: '',
                args: [''],
                chainId: 'mychannel',
                txId: tx_id
        };
```

Prvo dodamo koju funkciju želimo provesti, recimo da želimo zamijeniti automobil s nekim od korisnika.

```
                chaincodeId: 'fabcar',
                fcn: 'changeCarOwner',
                args: ['CAR7','Neven'],
                chainId: 'mychannel',
                txId: tx_id
```

Ovime smo dali do znanja da promjena koju želimo izvesti se odnosi na promjenu vlasništva automobila i da novi vlasnik automobila „CAR7“ postaje Neven. Spremanjem promjena i pokretanjem skripte dobivamo sljedeće:

```
Store path:/home/neven/fabric-samples/fabcar/hfc-key-store
Successfully loaded user1 from persistence
Assigning transaction_id:
debe5ef6ab4a3a67eea51f8bae59d6f2c93f00677e7420556c7a49dfdd4ad012
Transaction proposal was good
Successfully sent Proposal and received ProposalResponse: Status - 200,
message - ""
The transaction has been committed on peer localhost:7051
Send transaction promise and event listener promise have completed
Successfully sent transaction to the orderer.
Successfully committed the change to the ledger by the peer
```

Invoke.js skripta isto kao i query.js uzima autorizacijske podatke korisnika i komunicira s mrežom, odnosno s čvorom u mreži. Formulira naredbu za pozivanje funkcije koja pripada API-ju koji je dostupan fabacar pametnom ugovoru.

9. Zaključak

Ukoliko želimo privatan blockchain na kojem možemo dodavati organizacije i osobe koje imaju različite privilegije i dostupnosti, a ne želimo nepotrebno granati neki drugi blockchain, Hyperledger je odlično rješenje. Hyperledger dolazi s potpunim autorizacijskim sistemom i svakom korisniku mogu se pridružiti određene dozvole. Sadrži više mogućih mehanizama konsenzusa ovisno o stupnju povjerenja koje vlada između organizacija. Pošto svi čvorovi nemaju ista prava, odnosno neki imaju veća prava od drugih, to omogućuje odabir najboljeg mehanizma konsenzusa za trenutnu situaciju. Za razliku od Bitcoin-a ili Ethereum-a gdje se konsenzus mora ispuniti na cijelom blockchainu kod Hyperledgera konsenzus o ispravnosti transakcije vrši se na razini transakcije, odnosno transakcija mora ispasti validna kako bi se provela, ne ispituje se cijeli blockchain. Podržava već poznate i korištene programske jezike, kao što su Java, Nodejs i Go što ga čini puno pristupačnijim većem broju developera. Iako može generirati kriptovalute, one nisu potrebne za ispravan rad blockchaina, ne koristi Proof of Work niti Proof of Stake i ukoliko postoji u naprijed određeno povjerenje između organizacija omogućuje čak 1000+ transakcija u sekundi.

Nastavi li se razvijati tempom i u smjeru u kojem je krenuo Hyperledger bi mogao postati najbolje blockchain rješenje za B2B (Business to business) poslovanje, koje bi poduzećima uvelike olakšalo dogovaranje, omogućilo prilagođavanje na razne potrebe ovisno o poduzeću s kojim posluju, a uz to uvelike ubrzalo tijek transakcija i njihovu validaciju koja bi se vršila automatski. Na samome kraju financijski bi pripomoglo, pošto bi se troškovi transakcija uvelike smanjili.

10. Literatura

[1] Antonopoulous A.,M., (2017) Mastering Bitcoin. United States of America: O'Relly Media Inc.

[2] Bashir I., (2017) Mastering Blockchain. Birmingham UK: Packt Publishing.Ltd

[3] blockgeeks.com (2018) Smart Contracts: The Blockchain Technology That Will Replace

[4] dotmagazine.online (2018) Security and privacy in blockchain environments, preuzeto 18.08.2018. s https://www.dotmagazine.online/issues/innovation-in-digital-commerce/what-can-blockchain-do/security-and-privacy-in-blockchain-environments

[5] fortune.com (2018) How Should We Regulate Blockchain, preuzeto 19.08.2018. s http://fortune.com/2018/06/25/blockchain-cryptocurrency-technology-regulation-bitcoin-ethereum/

[6] hyperledger.org (2018) An Introduction to Hyperledger, preuzeto 21.08.2018. s https://www.hyperledger.org/wp-content/uploads/2018/08/HL_Whitepaper_IntroductiontoHyperledger.pdf

[7] hyperledger.org (2018) Hyperledger Architecture, Volume II, preuzeto 21.08.2018. s https://www.hyperledger.org/wp-content/uploads/2018/04/Hyperledger_Arch_WG_Paper_2_SmartContracts.pdf

[8] hyperledger-fabric.readthedocs.io (2018) A Blockchain Platform for the Enterprise, preuzeto 22.08.2018. s https://hyperledger-fabric.readthedocs.io/en/release-1.2/

[9] Ibm.com (2018) Service-oriented Architecture, preuzeto 19.08.2018. s https://www.ibm.com/support/knowledgecenter/en/SSMQ79_9.5.1/com.ibm.egl.pg.doc/topics/pegl_serv_overview.html

[10] Kozlovski S.,hackernoon.com (2018) A Thorough Introduction to Distributed Systems , preuzeto 22.08.2018. s https://hackernoon.com/a-thorough-introduction-to-distributed-systems-3b91562c9b3c

[11] Lawyers, preuzeto 19.08.2018. s https://blockgeeks.com/guides/smart-contracts/

[12] Medium.com (2018) The Oracle Problem, preuzeto 22.08.2018. s https://medium.com/@DelphiSystems/the-oracle-problem-856ccbdbd14f

[13] opengroup.org (2018) Service-Oriented Architecture Standards, preuzeto 19.08.2018. s http://www.opengroup.org/standards/soa

[14] opengroup.org (2018) What is SOA, preuzeto 19.08.2018. s https://web.archive.org/web/20160819141303/http://opengroup.org/soa/source-book/soa/soa.htm

[15] Peacock R. (2000.) Distributed Architecture Technologies, preuzeto 22.08.2018. s https://www.cp.eng.chula.ac.th/~fyta/684%20ISA/Papers/Peacock%20-%20dist%20arch%20tech.pdf
[16] Slobodnik J., Medium.com (2018) How Oracles connect Smart Contracts to the Real World, preuzeto 22.08.2018. s https://medium.com/bethereum/how-oracles-connect-smart-contracts-to-the-real-world-a56d3ed6a507
[17] Szabo N. (1996) Smart Contracts: Building Blocks for Digital Markets, preuzeto 22.08.2018. s http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html
[18] techopedia.com (2018) Peer-to-Peer Arhitecture, preuzeto 19.08.2018. s https://www.techopedia.com/definition/454/peer-to-peer-architecture-p2p-architecture
[19] techopedia.com (2018) Three-Tier Architecture, preuzeto 19.08.2018. s https://www.techopedia.com/definition/24649/three-tier-architecture
[20] techopedia.com (2018) Two-Tier Architecture, preuzeto 22.08.2018. s https://www.techopedia.com/definition/467/two-tier-architecture
[21] Techtarget.com (2018) Distributed applications preuzeto 22.08.2018. s https://searchitoperations.techtarget.com/definition/distributed-applications-distributed-apps

11. Popis slika