УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Aleksa Mirković

Blockchain distribuirani sistemi i njihova primena u farmaceutskoj industriji

Master rad

Novi Sad, 2017.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Мастер рад
Аутор, **АУ**:		Алекса Мирковић
Ментор, **МН**:		др Душан Гајић, доцент
Наслов рада, **НР**:		Blockchain дистрибуирани системи и њихова примена у фармацеутској индустрији
Језик публикације, **ЈП**:		Српски / латиница
Језик извода, **ЈИ**:		Српски
Земља публиковања, **ЗП**:		Република Србија
Уже географско подручје, **УГП**:		Војводина
Година, **ГО**:		2017.
Издавач, **ИЗ**:		Ауторски репринт
Место и адреса, **МА**:		Нови Сад, трг Доситеја Обрадовића 6
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		5/47/32/2/23/0/3
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Примењене рачунарске науке и информатика
Предметна одредница/Кључне речи, **ПО**:		Базе података и информациони системи, дистрибуирани системи, blockchain
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		
Датум прихватања теме, **ДП**:		18.9.2017.
Датум одбране, **ДО**:		4.10.2017.
Чланови комисије, **КО**:	Председник:	др Иван Луковић, редовни професор
	Члан:	др Силвиа Гилезан, редовни професор
	Члан, ментор:	др Душан Гајић, доцент

Потпис ментора

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Master Thesis	
Author, **AU**:		Aleksa Mirković	
Mentor, **MN**:		Dušan Gajić, Assistant Professor, Ph. D.	
Title, **TI**:		Blockchain Distributed Systems and Their Application in the Pharmaceutical Industry	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2017.	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovića sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		5/47/32/2/23/0/3	
Scientific field, **SF**:		Electrical and computer engineering	
Scientific discipline, **SD**:		Applied computer science and informatics	
Subject/Key words, **S/KW**:		Databases and information systems, distributed systems, blockchain	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad	
Note, **N**:			
Abstract, **AB**:			
Accepted by the Scientific Board on, **ASB**:		18/9/2017	
Defended on, **DE**:		4/10/2017	
Defended Board, **DB**:	President:	Ivan Luković, Full Professor, Ph. D.	
	Member:	Silvia Gilezan, Full Professor, Ph. D.	Menthor's sign
	Member, Mentor:	Dušan Gajić, Assistant Professor, Ph. D.	

	УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА** 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Датум:
	ЗАДАТАК ЗА МАСТЕР РАДА	Лист/Листова: 4/

(Податке уноси предметни наставник - ментор)

Студијски програм:	Рачунарство и аутоматика
Руководилац студијског програма:	др Мирослав Поповић, ред. проф.

Студент:	Алекса Мирковић	Број индекса:	Е2 57/2016
Област:	Електротехничко и рачунарско инжењерство		
Ментор:	др Душан Гајић, доцент		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;
- литература

НАСЛОВ МАСТЕР РАДА:

BLOCKCHAIN ДИСТРИБУИРАНИ СИСТЕМИ И ЊИХОВА ПРИМЕНА У ФАРМАЦЕУТСКОЈ ИНДУСТРИЈИ

ТЕКСТ ЗАДАТКА:

- Проучити основне концепте *blockchain* дистрибуираних система.
- Упознати се са *Hyperledger Fabric* платформом као примером конкретне имплементације *blockchain*-а.
- Упоредити *Hyperledger Fabric* и *Bitcoin* имплементације *blockchain* система.
- Упознати се са *Hyperledger Composer* алатом за развој пословних апликација базираних на *Hyperledger Fabric* платформи.
- Дефинисати и реализовати модел *blockchain* мреже намењене за подршку ланцу набавке у фармацеутској индустрији. Модел треба да садржи дефиниције ресурса, учесника и операција које учесници могу извршавати над ресурсима.
- На основу развијеног модела, имплементирати одговарајуће софтверско решење засновано на *Hyperledger Fabric*-u.

Руководилац студијског програма:	Ментор рада:
др Мирослав Поповић, ред. проф.	др Душан Гајић, доцент

Примерак за: ☐ - Студента; ☐ - Ментора

Sadržaj

1. Uvod 1
2. Osnove Blockchain distribuiranih sistema 3
   2.1. Komunikacioni model 4
   2.2. Dnevnik transakcija 6
   2.3. Konsenzus 8
      2.3.1. Dokaz posla 9
      2.3.2. Dokaz posedovanja 10
      2.3.3. Praktična vizantijska tolerancija greške 11
   2.4. Pametni ugovori 12
3. Hyperledger Fabric 13
   3.1. Projekat Hyperledger 13
   3.2. Hyperledger Fabric 13
      3.2.1. Vrste čvorova 14
      3.2.2. Smeštanje podataka 15
      3.2.3. Kanali 16
      3.2.4. Kod lanca 16
      3.2.5. Digitalna dobra 18
      3.2.6. Konsenzus 19
      3.2.7. Pružalac servisa članstva 19
      3.2.8. Tok transakcije 20
   3.3. Poređenje Hyperledger Fabric-a i Bitcoin-a 21
      3.3.1. Osvrt na Bitcoin 21
      3.3.2. Razlike Hyperledger Fabric-a i Bitcoin-a 24
4. Primena blockchain distribuiranih sistema u oblasti farmacije 26
   4.1. Fabric Composer 26
   4.2. Opis rešenja 28
      4.2.1. Opis procesa 29

4.2.2. Opis implementacije 29
5. Zaključak 40
Literatura 42
Prilog 1: Spisak korišćenih termina 44
Prilog 2: Spisak tabela 45
Prilog 3: Spisak slika 46
Biografija 47

1. Uvod

Razmena dobara i usluga, kao aktivnost, prisutna je još od pojave društva, a svaka konkretna razmena predstavlja transakciju. U poslovanju, ona je jedinica kojom se ispunjavaju ugovorne obaveze. Povećanje obima poslovanja nametnulo je potrebu beleženja i čuvanja izvršenih transakcija u cilju adekvatnog upravljanja poslovnim procesima. Nekadašnje čuvanje podataka na papirima potisnuo je napredak tehnologije i za taj zadatak uposlio računar.

Prvobitno su se računarski informacioni sistemi oslanjali na sistem datoteka za skladištenje podataka. Kod sistema datoteka bilo je komplikovano implementirati izvršenje poslovne transakcije kao niza operacija upisa i čitanja koja ili u celosti uspe ili u celosti ne uspe. Krajem dvadesetog veka relacione baze podataka postaju nezaobilazni činilac informacionih sistema [31]. Oslanjajući se na sistem za upravljanje bazom podataka, relaciona baza podataka podržava izvršavanje poslovne transakcija kao nedeljivog niza operacija nad bazom podataka. Evidentiranje transakcija omogućuje analizu poslovanja. Sa stanovišta načina skladištenja podataka delimo baze podataka na centralizovane i distribuirane.

Centralizovane baze podrazumevaju smeštanje podataka na jednu lokaciju. Kod njih je redundansa podataka minimalna. Ovakve baze su znatno jednostavnije za upravljanje od distribuiranih.

Usled velikog rasta količine podataka koji se nalaze širom Interneta, u prethodnih nekoliko godina distribuirane baze postaju sve popularnije. Kod distribiranih baza se može iskoristiti memorijski kapacitet i procesorska moć svih računara na kojima je baza distrbuirana. Ako se podaci izgube u centralizovanoj bazi, oporavak je najčešće vrlo težak i ceo sistem postaje nedostupan. Kod distribuiranih sistema, podaci mogu biti dostupni na više lokacija, pa gubitak podataka sa jedne lokacije nije fatalan.

Distribuirane baze pružaju veću stabilnost i dostupnost sistema, ali ne rešavaju neke probleme načina poslovanja među partnerima. Prilikom saradnje svaki učesnik beleži sopstvene podatke i može doći do razlika u podacima koji se tiču međusobnog poslovanja. Neslaganje u podacima nastaje usled greške ili pokušaja prevare. To stvara nepoverenje među učesnicima u poslovanju i često zahteva uvođenje posrednika. Banka se smatra nezaobilaznim posrednikom prilikom poslovanja dva entiteta. Bankarske usluge često mogu biti spore, a kompanije plaćaju veliki novac za te usluge. Blockchain sistemi omogućavaju da se poslovanje odvija bez posrednika, a da se zadrže prednosti koje pružaju distribuirane baze. Blockchain koncept je prvi put uveden kao tehnologija koja će omogućiti rad kriptovalute (digitalno sredstvo za razmenu) Bitcoin [1]. Primena tehnologija zasnovanih na blockchain-u (engl. *blockchain - lanac blokova*) u poslovnim sistemima bi trebalo da doprinese većoj transparentnosti, većoj sigurnosti, većem otporu na otkaze, većoj brzini izvšavanja transakcija i smanjenoj ceni troškova transakcija usled uklanjanja posrednika [1]. Korišćenjem blockchain-a svi učesnici zajedno upravljaju podacima

relevantnim za njihovo poslovanje, nasuprot tradicionalnim sistemima gde svaka strana održava sopstvene podatke.

Cilj ovog rada je da predloži način korišćenja blockchain tehnologije u farmaceutskoj oblasti. Potrebno je obraditi aspekt naručivanja, odnosno dostavljanja farmaceutskih proizvoda i ispratiti taj proces od proizvođača do krajnjih korisnika. Među kompanijama koje se bave ovom delatnošću informacije se uglavnom prenose elektronskom razmenom dokumenata, odnosno već postoji dobra tehnička osnova. Blockchain sisteme je pogodno primeniti u ovom domenu jer postoji veliki broj učesnika u lancu i veliki broj transakcija. U ovoj industriji je veliki obrt novca i mnogo prostora za malverzacije. Nepostupanje u skladu sa pravilima može rezultirati narušavanjem ljudskog zdravlja usled konzumacije ilegalnih lekova. Blockchain tehnologija bi trebalo da obezbedi poštovanje unapred definisanih pravila među poslovnim akterima.

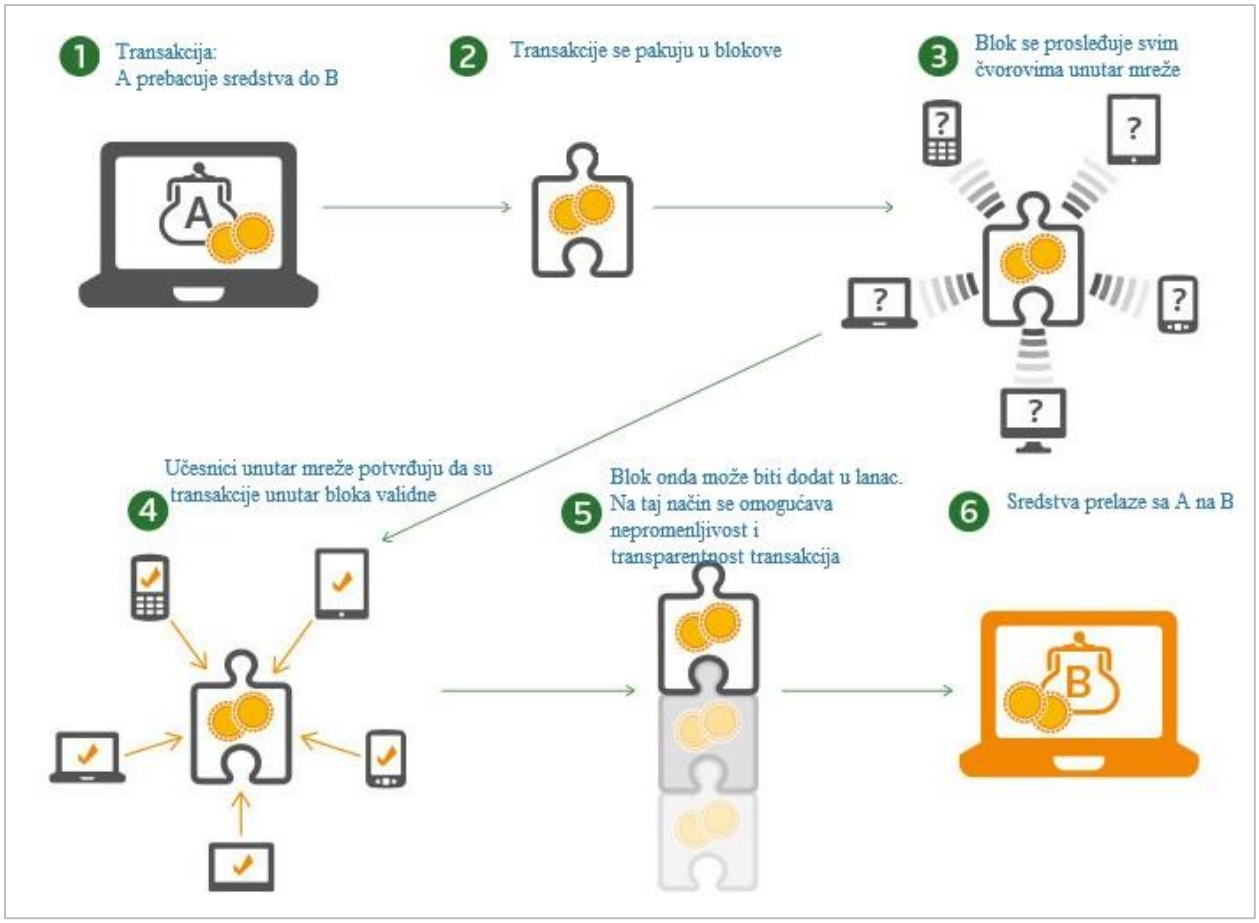
Na osnovu postavljenog cilja definisano je šest zadatka. Prvi zadatak podrazumeva izučavanje osnovnih koncepata blochchain distribuiranih sistema. Drugi zadatak se tiče upoznavanja sa Hyperledger Fabric [7] platformom kao primerom konkretne implementacije blockchain-a, a treći zadatak podrazumeva poređenje Hyperledger Fabric i Bitcoin implementacije blockchain sistema. Četvrti zadatak obuhvata upoznavanje sa Hyperledger Composer alatom [4]. Peti zadatak podrazumeva definisanje modela blockchain mreže namenjene za podršku lancu nabavke u farmaceutskoj industriji. Za realizaciju šestog zadatka, potrebno je implementirati softversko rešenje na osnovu razvijenog modela.

Ostatak ovog master rada ima sledeću strukturu. U drugom poglavlju, pod nazivom „Osnove blockchain distribuiranih sistema”, detaljno je objašnjen princip rada blockchain tehnologije, komunikacija unutar blockchain mreže, način ulančavanja blokova, konsenzus i pametni ugovori. Treće poglavlje razmatra Hyperledger Fabric i potom se bavi poređenjem implementacije Hyperledgera Fabric-a i Bitcoin-a. Praktični deo rada je opisan u četvrtom poglavlju pod nazivom „Primena blockchain distribuiranih sistema u oblasti farmacije“. U tom poglavlju je opisana implementacija funkcionalnosti za slučaj upotrebe blockchain-a u farmaceutskoj industriji. U petom poglavlju je dat zaključak gde se predlažu dalji pravci razvoja. Prilog 1, „Opis korišćenih termina” sadrži objašnjenja za pojmove koji se pojavljuju u ovom radu. Spisak tabela i slika je dat u prilogu 2 i prilogu 3, respektivno.

2. Osnove Blockchain distribuiranih sistema

Struktura koju danas nazivamo blockchain prvi put je uvedena u radu Satoshi Nakamoto-a iz 2008. godine [1]. Sličan način uvezivanja podataka predložen je i u radu Haber-a i Stornetta iz 1991. godine [11].

U najjednostavnijoj formulaciji se može reći da je blockchain deljeni, distribuirani, replicirani dnevnik transakcija (engl. *ledger*) [32]. Termin *ledger* označava glavnu knjigu u knjigovodstvu. Pomenuti dnevnik transakcija se u širem smislu može posmatrati kao baza podataka. Ipak, dnevnik je samo jedna komponenta blockchain tehnologije. Blockchain tehnologija je drugačija od tradicionalnih baza sa strane potvrda transakcija. Kod tradicionalnih načina čuvanja podataka, svaka strana koja učestvuje u poslovanju ima svoje podatke. Kako bi poslovni partneri usaglasili podatke oni razmenjuju dokumenta u papirnom i audio obliku što može biti nepouzdano i neefikasno. Blockchain princip rada omogućava da svi učesnici poseduju iste podatke kojima svi učesnici veruju. Na osnovu pravila koja su utvrđena (konsenzusa) između članova mreže, transakcije se prihvataju ili odbijaju. Kada se transakcija jednom unese u dnevnik transakcija, ne postoji način da bude izbrisana. Tok transakcije prikazan je na slici 1 [34].



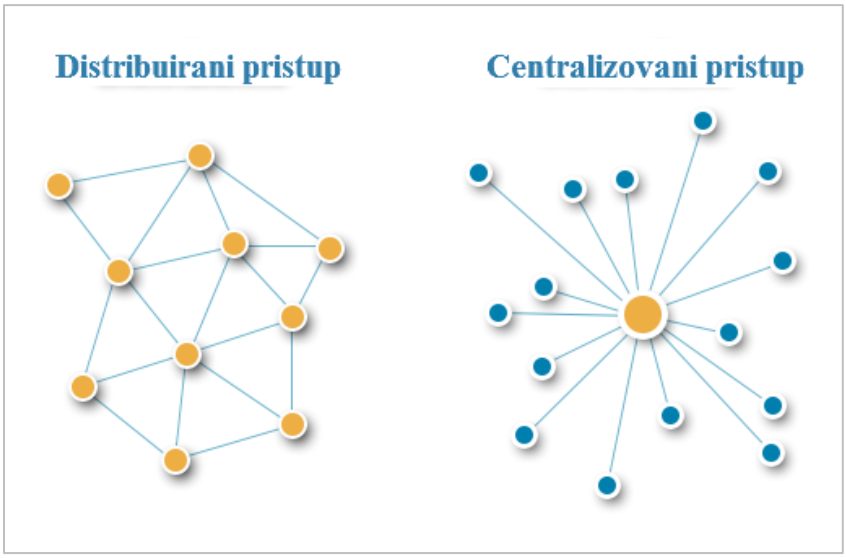
Slika 1. Propagacija transakcije kroz blockchain

Najvažniji činioci blockchain-a[1] su distribuirani dnevnik transakcija (baza podataka), pametni ugovori (engl. *smart contracts*) [9] i konsenzus [7]. Blockhain doprinosi većoj sigurnosti podataka, transparentnosti, uklanjanju posrednika, većoj brzini potvrde transakcija, većoj dostupnosti podataka i većem otporu na otkaze.

Kod tradicionalnih, centralizovanih sistema otkaz računara na kom se nalazi baza podataka znači gubitak svih podataka u sistemu. Za takve sisteme se kaže da imaju jednu tačku otkaza (engl. *Single Point of Failure - SPOF*). Blockhain tehnologija nameće da svaki čvor unutar mreže poseduje sopstvenu kopiju svih podataka.

2.1. Komunikacioni model

Komunikacija između aktera se odvija kroz peer-to-peer mrežu [1]. Svi čvorovi unutar mreže poseduju identične podatke. Svaki učesnik prosleđuje podatke do svih čvorova sa kojima je povezan. Kako se radi o distribuiranom sistemu, potrebno je posebnu pažnju posvetiti sinhronizaciji podataka. Na slici 2. prikazani su distribuirani i centralizovani načini organizacije [33].



Slika 2. Distribuirani i centralizovani pristup

Razlikujemo privatne i javne blockchain mreže. Prema [12] postoji i treća grupa koja se zove polu-privatna ili hibridna. Privatne mreže nazivamo zatvorenim, ili sa dozvolom, dok javne nazivamo otvorenim, javnim ili bez dozvole.

[1] Nekad se pod nazivom blockchain misli samo na dnevnik transakcija, dok u ovom radu, kada se kaže blockchain misli se na blockchain tehnologiju, koja obuhvata specifičan način rada

Javne mreže su takve da svako može da se pridruži, ima prava upisa u bazu podataka, kao i čitanja iz nje. Svi čvorovi unutar ovakvih mreža mogu da učestvuju u potvrdi transakcija, najčešće kroz dokaz posla (engl. *Proof of Work*) ili dokaz posedovanja (engl. *Proof of Stake*) sisteme. Ovakvi sistemi potvrde su skupi sa stanovišta vremena i novca, ali osiguravaju ispravnost mreže. Učesnici unutar mreže su anonimni. Najpoznatija javna mreža je Bitcoin.

Nasuprot njima, privatne mreže podrazumevaju da postoje različita prava pristupa za čvorove unutar mreže. Identiteti učesnika u mreži su poznati i na osnovu identiteta su im dodeljena prava pristupa. Kako se ovakve mreže mogu primeniti u poslovnom okruženju, brzina prihvatanja transakcije je veoma bitna, pa se se konsenzus postiže drugačijim mehanizmima nego u javnim mrežama. Razlike između privatnih i javnih mreža su date u tabeli 1.

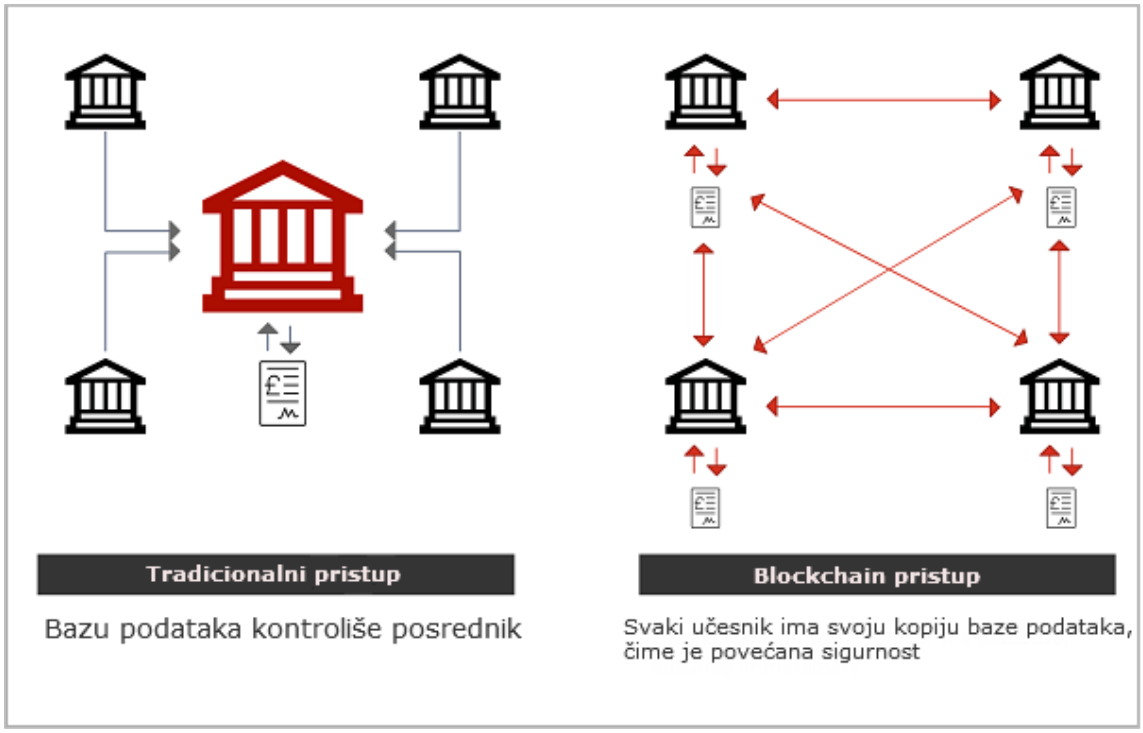
	Javna mreža	Privatna mreža
Pristup	Otvoreni pristup za čitanje i pisanje u bazu	Čitanje i pisanje u bazu zahteva dozvolu
Potvrda transakcije	Sporije	Brže
Održavanje ispravnosti mreže	Dokaz posla/dokaz posedovanja	Unapred određeni učesnici
Identitet učesnika	Anonimni	Identifikovani
Održavanje	Skupo	Jeftino
Broj čvorova	Milioni	Do nekoliko stotina
Primer	Bitcoin	Hyperledger

Tabela 1. Razlike između privatnih i javnih blockchain mreža

Pre nego što transakcija može biti upisana u ledger, ona je kreirana od strane nekog čvora (učesnika unutar mreže). Nakon toga, transakcija se propagira do drugih čvorova unutar mreže, sa tim da se vodi računa koje transakcije su već propagirane do susednih čvorova. Određeni čvorovi koji se još nazivaju i rudari[2] zatim grupišu transakcije u blok i šalju ih ostatku mreže na verifikaciju, kako bi se kasnije blokovi dodali u dnevnik transakcija. Može se desiti da usled vremena potrebnog da se sinhronizuje baza dođe do razilaženja dela mreže, tj. da jedan deo mreže smatra jedan lanac blokova validnim, dok drugi smatra drugačiji lanac ispravnim. Ovakva pojava se naziva viljuška (engl. *fork)* [20].

Blockchain tehnologija je karakteristična po tome što svi učesnici mreže posmatraju iste podatke. Najbitnija karakteristika ove tehnologije je potpuno ukidanje ili smanjenje uloge posrednika prilikom poslovanja između učesnika unutar mreže, što je prikazano na slici 3 [35].

[2] Rudarenje je proces dodavanja transakcija (u vidu bloka) na kraj lanca blokova. Rudari su čvorovi koji se bave rudarenjem.



Slika 3. Tradicionalni i blockchain pristup

Uklanjanjem posrednika smanjuje se vreme potrebno da se neka transakcija izvrši. Osim toga smanjuju se i novčani troškovi. Kod blockchain pristupa svi poseduju sopstvenu kopiju podataka kojima veruju. Za svaku promenu nad podacima, saglasnost moraju dati relevantne strane (konsenzus). Ukoliko neki čvor postane nedostupan, drugi čvorovi imaju kopiju podataka. Nasuprot blockchain distribuiranim sistemima, otkaz računara koji sadrži podatke kod centralizovanih sistema je često predstavljao nerešiv problem.

2.2. Dnevnik transakcija

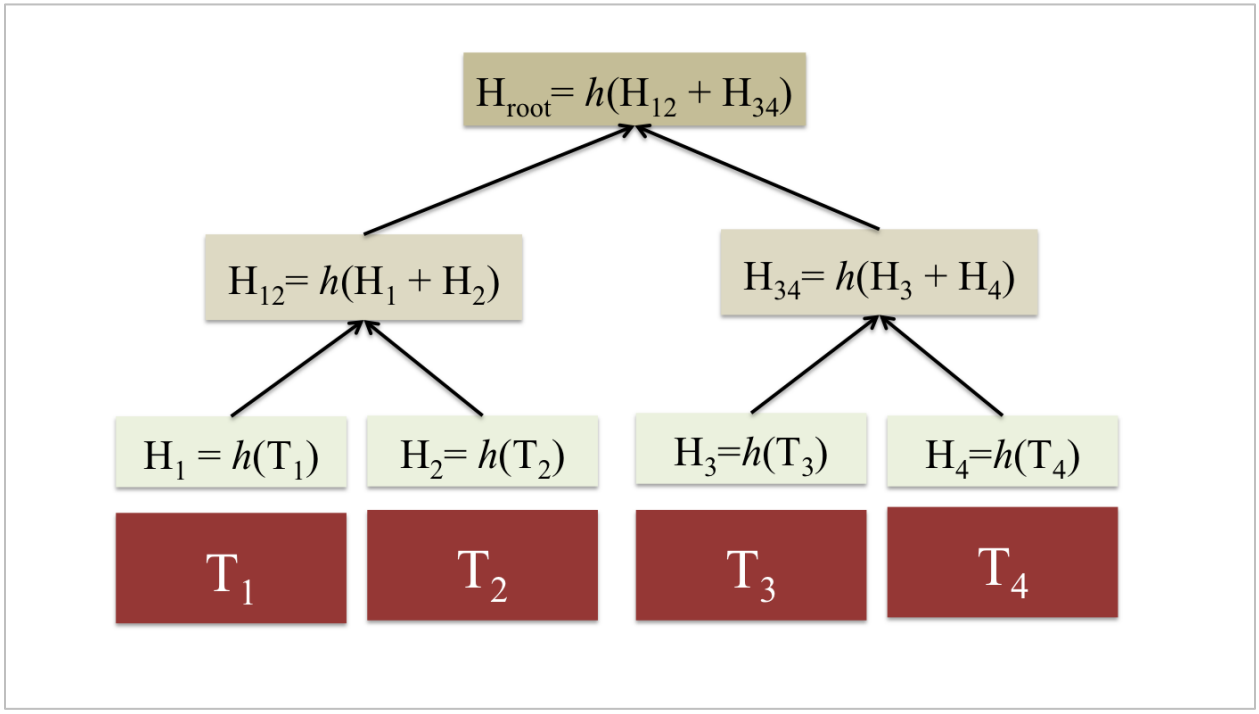
Dnevnik transakcija je struktura koja se koristi za smeštanje transakcija u blockchain sistemima. Blokovi koji sadrže potvrđene transakcije se u linearnom poretku uvezuju i odatle naziv blockchain. Svaka validna transakcija se u sklopu odgovarajućeg bloka upisuje u dnevnik transakcija i zauvek ostaje tu. Za razliku od tradicionalnih baza podataka, ne postoji način da se nešto što je upisano u ovakvu bazu podataka (dnevnik transakcija) izmeni. U slučaju da transakcija koja je upisana nije u skladu sa realnim sistemom, potrebno je napraviti novu transakciju koja će dovesti bazu podataka u stanje konzistentnosti sa realnim stanjem.

Struktura bloka nije ista za svaku implementaciju blockchain-a. Ono što je zajedničko za svaku implementaciju je postojanje sekcije podataka bloka i zaglavlja bloka. Zaglavlje bloka referencira zaglavlje prethodnog bloka. Svaki blok kao jedan od atributa u svom zaglavlju sadrži heš vrednost, odnosno heš (engl. *hash*) svog prethodnika. Osim zaglavlja postoji i sekcija podataka bloka gde se nalaze transakcije. Heš bloka se računa na osnovu korisnih podataka

(transakcija) unutar bloka i heša prethodnog bloka. U zavisnosti od implementacije ulazni podaci za heš funkciju mogu biti prošireni. Bitcoin implementacija nameće da zaglavlje bloka sadrži i timestamp kao i *nonce*, odnosno broj koji se koristi kao potvrda da je čvor uložio procesorsko vreme da validira blok transakcija.

Kao što je rečeno blok sadrži sekciju podataka i zaglavlje. Sekcija podataka u osnovnom obliku sadrži transakcije, ali može sadržati i bilo kakve druge podatke. Sadržaj sekcije podataka se hešira[3] i zatim se dobijeni rezultat smešta unutar zaglavlja bloka.

Merkle stablo [5] je struktura stabla koja se koristi za heširanje sekcije podataka bloka. Ovakve strukture se i u drugim sistemima koriste za verifikaciju podataka koji se šalju kroz mrežu. Prema [5] da bi se formiralo Merkle stablo, prvo se svaka transakcija iz sekcije podataka hešira čime nastaju listovi stabla. Na sledećem nivou se heširaju heševi koji se nalaze u listovima stabla. Na primer, na slici 4 [36], H_{12} je dobijen heširanjem stringa koji je dobijen konkatenacijom heševa H_1 i H_2.



Slika 4. Merkle stablo

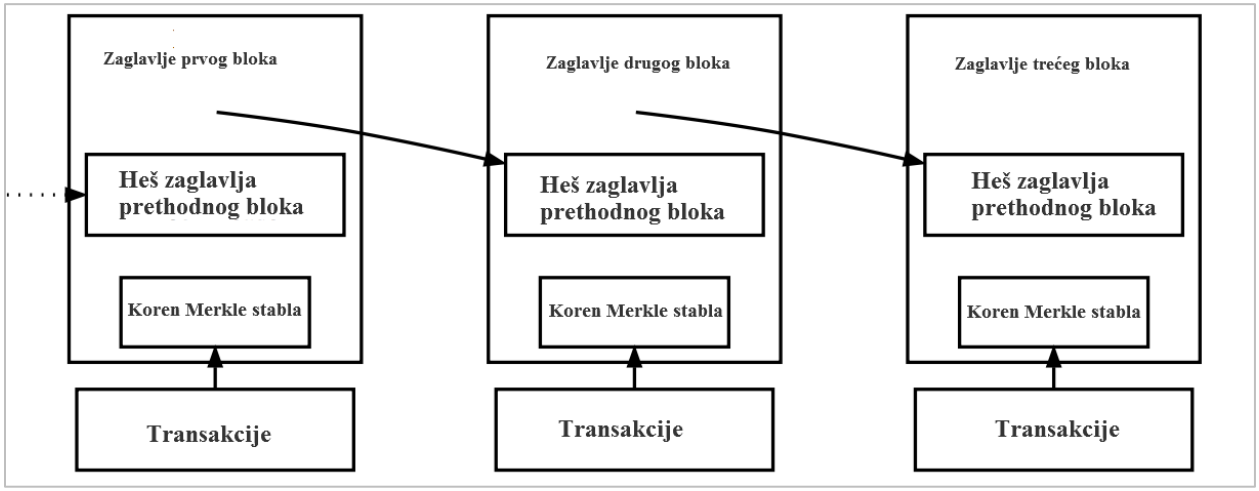
Postupak se nastavlja sve dok se ne dođe do korenskog heša odnosno korena Merkle stabla (H_{root}). Heširanje se najčešće vrši koristeći SHA2 algoritam [22]. Heš koji se nalazi u korenu stabla predstavlja heširan sadržaj svih podataka unutar sekcije podataka.

Merkle stabla se uglavnom implementiraju kao binarna. Sa stanovišta performansi, za skup od n podataka (transakcija) potrebno je izvršiti oko $2n$ heširanja. U slučaju da je broj čvorova u stablu u nekom nivou neparan, onda se poslednji čvor sa leve strane u datom nivou

[3] Heširati – trasformisati ulaz heš funkcijom

hešira 2 puta, kako bi se dobio element koji mu odgovara na nivou iznad. Provera da li se neki element nalazi u skupu zahteva da su dostupni svi elementi na putu od traženog elementa do korena. To zahteva log(*n*) heširanja.

Na opisani način dobija se povezana struktura blokova koja je otporna na izmene. Promena transakcije u jednom bloku bi rezultirala drugačijim hešom tog bloka. Pošto heš tog bloka učestvuje kao ulazni podatak za heš narednog bloka, znači da će ostatak mreže videti ako je menjan lanac. Izmena sadržaja ili heša jednog bloka će uzrokovati da svi heševi blokova u lancu nakon njega budu promenjeni, a takvu aktivnost bi ostatak mreže najverovatnije prepoznao kao malicioznu i odbacio. Lanac blokova je prikazan na slici 5 [38].



Slika 5. Lanac blokova

2.3. Konsenzus

Proces održavanja dnevnika transakcija sinhronizovanog preko mreže naziva se konsenzusom [3]. On podrazumeva osiguravanje ažuriranja dnevnika transakcija isključivo kada su transakcije odobrene od strane odgovarajućih učesnika, a da kada se dnevnik ažurira to kod svih čvorova bude sa istim transakcijama u istom redosledu.

Kod tradicionalnog pristupa i poslovanja među poslovnim subjektima, postoji problem poverenja i utvrđivanja koji podaci su validni. Neki poslovni entiteti mogu imati drugačije podatke od ostatka poslovnih subjekata sa kojima sarađuju, zbog greške ili zbog pokušaja prevare. Takvi problemi su se ranije rešavali uz pomoć jednog ili više posrednika. Takvo rešenje je skupo i vremenski zahtevno. Blockchain pruža rešenje tako da nije potreban posrednik. U ovu svrhu se implementiraju pravila za prihvatanje transakcija u ledger.

Postoje mnogobrojni sistemi za postizanje konsenzusa u blockchain mrežama, a ovde će biti detaljno opisani dokaz posla, dokaz posedovanja i praktična vizantijska tolerancija greške (engl. *practical byzantine fault tolerance*).

2.3.1. Dokaz posla

Jedan od načina da se javna blockchain mreža održi sigurnom je korišćenjem protokola koji se naziva dokaz posla. Ideja je da čvor unutar mreže mora uložiti neku količinu procesorskog vremena da bi poslao podatke ostatku mreže na verifikaciju. Odlika sistema je da čvor koji šalje poruku mora da reši kompleksan računarski problem. Rešenje tog problema se pridružuje poruci koja se šalje, a za ostatak mreže je veoma jednostavno da proveri da li je problem rešen ispravno.

Bitcoin koristi dokaz posla protokol da bi postigao konsenzus, preciznije koristi HashCash. Ovaj sistem je prvi put opisan u [14] kao rešenje za problem *e-mail* spamova.

Pre nego što se pošalje mejl, pošiljalac je morao da digitalno označi *e-mail* poruku. Digitalna oznaka se smešta u zaglavlje *e-mail* poruke. Oznaka se dobija transformacijom tj. heširanjem ulaznih podataka (atributa poruke). Pomenuti ulazni podaci su vreme slanja poruke, adresa elektronske pošte primaoca i *nonce*. Kako je u to vreme korišćen SHA1 algoritam koji koristi 160 bita za enkripciju, znači da postoji 2^{160} različitih heševa. Da bi *e-mail* poruka imala validan potpis zahtevalo se da npr. prvih 20 bita budu 0, što znači da je broj validnih heševa 2^{140}, odnosno da bi bilo potrebno u proseku 2^{20} pokušaja da se nađe odgovarajući heš. Kako se u ulaznim podacima nalazi i nonce, potrebno je pronaći takav nonce tako da rezultujući heš ispunjava pomenute kriterijume. Ne postoji način da se nonce pogodi osim isprobavanja svih kombinacija (engl. *brute force*). To dalje implicira da će računar pošiljalac morati da uloži procesorsku moć i vreme kako bi se heš, odnosno nonce pronašao. Spameri čiji je posao da šalju veliki broj poruka bi morali pre svakog slanja da potroše dodatno vreme da izračunaju ispravan heš. Sa druge strane, za primaoca elektronske poruke je vrlo jednostavno da izračuna heš nad podacima u zaglavlju elektronske poruke i proveri da li heš ispunjava kriterijume. Pored toga, vrše se provere kao što je provera validnosti adrese elektronske pošte pošiljaoca i provera da li je u prošlosti stigla poruka sa istim hešom.

HashCash koristi SHA256 algoritam čiji se opis može naći u [22]. Bitcoin koristi implementaciju HashCash algoritma takvu da zahteva da heš koji je validan mora da bude manji od zadate vrednosti. Što je zadata vrednost manja, to je teže naći odgovarajući heš. Na svakih 2016 blokova koji se uvežu u Bitcoin blockchain, menja se zadata vrednost, tako da zadovolji kriterijum da se novi blok uveže u lanac za prosečno 10 minuta [24].

Svaka heš funkcija je jednosmerna, odnosno nema sebi inverznu, ili je vrlo teško naći je. Ako obeležimo funkciju heširanja slovom H, ulaz u nju sa x i izlaz sa y, trebalo bi da bude jednostavno naći $y = H(x)$, ali veoma teško pronaći funkciju I, takvu da je $I(y) = x$. Pored toga, bitno svojstvo heš funkcija je da se kolizije između heševa ne dešavaju često. Zapravo, bitan je kardinalitet skupa slika heš funkcija, jer koliko god taj skup bio veliki, on je konačan. Za razliku od izlaza, broj ulaza je beskonačan i sigurno da postoje ulazi koji će dati iste izlaze. Slična argumentacija je data u [20]. Heš funckije su determinističke i za isti ulaz daju isti izlaz svaki put.

SHA256 na izlazu daje heš fiksne dužine. Poželjna osobina funkcija heširanja je da i mala promena ulaza daje potpuno drugačiji izlaz. Ova pojava se zove efekat lavine (engl. *avalanche effect*). To prema [23] znači da ako se makar jedan bit na ulazu promeni, svaki bit na izlazu će se promeniti sa verovatnoćom 0.5. Ovu osobinu poseduje i SHA256, što se može videti na slici 6.

```
S1=Kompjuter
S2=Kompjuter1
SHA256(S1)= ed2398e8c57f0d87ebda8921ec9a911a9e250e2e3118fbeeee288753d0924488
SHA256(S2)= d5e3586028e3acdd5cffc9b3f0d5d2593666520daacf4b22311f4629119c72bb
```

Slika 6. Male promene na ulazu u SHA256 rezultuju značajno različitim izlazom

Prvi čvor koji pruži dokaz posla smestiće novi blok na kraj lanca [1]. Kako se najčešće dokaz posla koristi kod javnih mreža koje se bave razmenom kriptovaluta, onda čvor koji doda blok na lanac dobija nagradu, odnosno neku količinu kriptovalute. Čvorovi koji računaju dokaz posla su rudari. Kod Bitcoina je upravo rudarenje jedini način nastajanja nove vrednosti odnosno stvaranja Bitcoin-a. S obzirom da svaki blok osim početnog bloka (engl. *genesis block*), za računanje heša koristi heš prethodnog bloka, to znači da što je duži lanac blokova, manja je verovatnoća da lanac bude kompromitovan. Da bi neko ugrozio potrebno je da poseduje 51% komjuterske moći cele mreže. Ovakvi napadi se zovu "51% napadi" ili napadi većine [25].

2.3.2. Dokaz posedovanja

Dokaz posedovanja (engl. *Proof of Stake*) predstavlja alternativu dokazu posla i prvi put je uveden u PeerCoinu [15]. Kao i dokaz posla, koristi se za postizanje konsenzusa u javnim blockchain mrežama.

Za razliku od dokaz posla sistema, dokaz posedovanja ne nameće takmičenje među čvorovima rudarima. Čvorovi koji uvezuju i potvrđuju blokove zovu se validatori. Svaki čvor koji želi da validira blok, mora da uloži neku količinu kriptovalute. Mreža na slučajan način bira validatora uzimajući u obzir količinu kriptovalute koju čvorovi poseduju.

Prednosti u odnosu na dokaz posla su sledeće:

- smanjenje troškova za opremu koju čvorovi rudari nabavljaju, odnosno skupocena oprema nije potrebna,
- štednja energije i manje zagađivanje sredine,
- veća zaštićenost od 51% napada; za razliku od dokaza posla gde bi napadač trebalo da poseduje 51% hardverske snage cele mreže, ovde je neophodno posedovati 51% količine kriptovalute da bi se izvršio uspešan napad.

Prema [3] se smatra da je komplikovanije da zlonamerni čvorovi poseduju 51% kriptovalute nego 51% procesorske moći kao što je slučaj kod dokaza posla. Da bi neko posedovao 51% kriptovalute potrebno je da uloži velika sredstva za tako nešto. Ukoliko bi neko zaista i posedovao 51% kriptovalute unutar mreže onda bi kompromitovanjem iste bio u velikom gubitku usled pada vrednosti kriptovalute.

Implementacija koja podrazumeva da se isključivo na osnovu količine kriptovalute koju čvorovi poseduju biraju validatori, najbogatijim čvorovima bi se davala nedostižna prednost i stvarala neželjena centralizovanost. Iz tog razloga postoje razne implementacije dokaza posedovanja. Jedna od varijanti podrazumeva da je osim količine kriptovalute bitno i koliko dugo je kriptovaluta u posedu korisnika. Što je duže u posedu i što je veća količina to je veća verovatnoća da će čvor biti izabran da ulanča novi blok. Nakon korišćenja kriptovalute njihova „starost“ se restartuje. Posle nekog perioda u posedu, „starost“ kriptovalute više nema značaja. Na ovaj način se izbegava mogućnost stvaranja velike količine valute kod male grupe čvorova. U osnovnoj implementaciji dokaza posla čvorovi validatori profitiraju isključivo kroz provizije[4] od transakcija koje validiraju.

2.3.3. Praktična vizantijska tolerancija greške

Praktična vizantijska tolerancija greške predstavlja jedan od algoritama kojim se rešava poznati problem vizantijskih generala [8]. Problem ukratko glasi:

Nekoliko generala sa vojskama nalazi se sa različitih strana neprijateljskog utvrđenja. Potrebno je da se dogovore oko toga da li će napasti utvrđenje i u koje vreme, jer samo ako dovoljan broj generala započne napad u isto vreme, neprijtalejski grad može biti osvojen. Postoji problem komunikacije, stoga generali šalju glasnike kako bi se dogovorili. Međutim, glasnici moraju da prođu pored neprijatelja, tako da oni mogu biti uhvaćeni, pa poruke možda nikada neće stići ili čak glasnici mogu biti kompromitovani pa dostavljati pogrešne poruke.

Preslikano na problem blockchaina postavlja se pitanje kako garantovati ispravnost podataka unutar mreže. Kod javnih blockchain mreža za tu svrhu se koristi dokaz posla. Kod privatnih mreža kao što je Hyperledger Fabric, potrebno je smisliti drugačiji algoritam usled manjeg broja učesnika u privatnoj mreži. Takođe, potrebno je da implementirani algoritam vrši bržu potvrdu transakcije.

Model problema je takav da mreža može biti nepouzdana, neki čvorovi mogu biti nedostupni ili slati pogrešne podatke koje ne moraju stizati redosledom kojim su poslate.

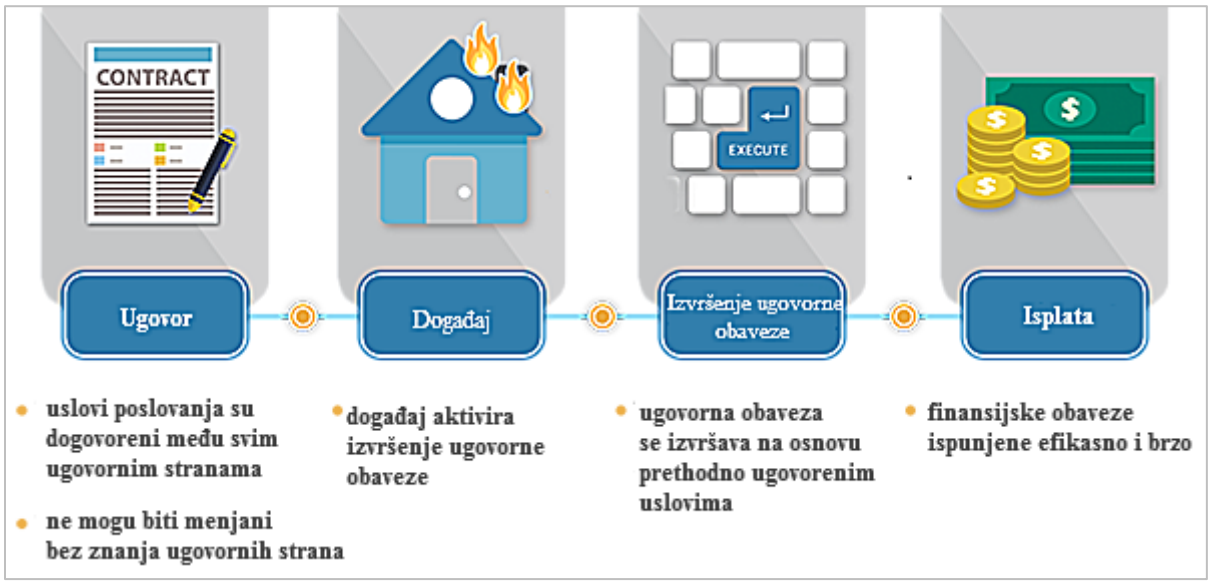
Castro i Liskov u [6] predlažu algoritam koji zahteva da ukupan broj čvorova u mreži bude minimalno $3f+1$, gde je f maksimalni broj neispravnih čvorova. Svaki čvor će sam za sebe odlučiti da li je poruka korektna, odnosno da li je neka transakcija ili niz transakcija u skladu sa pravilima poslovanja unutar mreže. Nakon toga, da bi poruka bila prihvaćena, odnosno transakcije bile sačuvane u dnevnik transakcija, potrebno je da $2f+1$ čvorova potvrdi tu radnju.

[4] Provizija od transakcija je deo vrednosti koja se prenosi transakcijom i ona ide čvoru koji uspešno potvrdi blok

2.4. Pametni ugovori

Prilikom bilo kakve saradnje između dva entiteta postoji neki utvrđeni dogovor. Najčešće su takvi vidovi saradnje regulisani ugovorom. U ugovoru su navedene obaveze, koje svaka strana koja je potpisnica dokumenta (ugovora) mora da ispuni. Blockchain podržava ovakve odnose među akterima mreže korišćenjem pametnih ugovora.

Pametni ugovori se koriste da bi se podržali oblici saradnje, odnosno poslovne logike koja je kompleksnija od jednostavne razmene dobara putem transakcija [9]. Prvi put je pojam pametnog ugovora uveden u [19]. Pametni ugovori su poslovni ugovori implementirani u programskom kodu. Oni se baš kao i pravni ugovori vode protokolom ako-onda (engl. *if-then*). To znači da pametni ugovori mogu automatski da obezbede ispunjenje uslova saradnje između entiteta. Način funkcionisanja pametnih ugovora prikazan je na slici 7 [37].



Slika 7. Princip rada pametnog ugovora

U slučaju da je učesnik A obavezan da dostavi robu učesniku B, pametni ugovor će osigurati da sredstva sa računa učesnika B ne pređu na račun učesnika A, sve dok dogovorena roba ne bude dostavljena učesniku B. Pored toga, osiguraće i da čim je roba stigla na dogovoreno odredište sredsva pređu na račun učesnika A. Ovakav sistem implicira automatsko plaćanje penala u slučaju kršenja pravila koja su propisana ugovorom. Takođe, svi akteri u mreži mogu da vide da li je neki učesnik ispunio svoje obaveze iz pametnog ugovora. Pametni ugovori su dostupni svim ugovornim stranama.

3. Hyperledger Fabric

Iako javne mreže imaju svojih prednosti, one nisu pogodne kada je potrebno implementirati poslovne sisteme. Kada slučaj korišćenja zahteva veći nivo privatnosti i sigurnosti, potrebno je koristiti privatne blockchain mreže. Javnim mrežama je nekada potrebno netoleranto mnogo vremena za potvrdu transakcije, a kod poslovnih sistema se nekad traži da vreme prihvatanja transakcije bude u intervalu od nekoliko sekundi. Zbog potreba da se blockchain prilagodi industriji kreiran je Hyperledger projekat [13].

3.1. Projekat Hyperledger

Hyperledger je kolaborativni projekat otvorenog koda stvoren sa ciljem da se blockchain tehnologija unapredi i primeni u različitim sektorima industrije. Projekat je pokrenut 2015. godine, mada se tek početkom 2016. godine krenulo sa aktivnim razvojem [13]. Nosilac projekta je Linux fondacija, a takođe u projektu učestvuju vodeće kompanije iz sveta finansija i tehnologije. Hyperledger obuhvata veliki broj platformi za razvoj kao što su Fabric, Sawtooth, Iroha i Burrow. Pored pomenutih platformi postoje i alati za razvoj. Najpoznatiji alati su Cello, Explorer i Composer.

3.2. Hyperledger Fabric

Hyperledger Fabric [7] je platforma za razvoj distribuiranih rešenja koja se oslanja na blockchain tehnologiju. Ova platforma ima modularnu arhitekturu koja je dizajnirana tako da podrži različite implementacije komponenata, kako bi se prilagodila raznim potrebama poslovanja. Osim toga, nudi i visok nivo skalabilnosti, a obezbeđuje sigurnost odnosno privatnost transakcija koja je neophodna u poslovnom okruženju.

Fabric verzija 1.0 je objavljena tek sredinom 2017. godine. To samo govori o tome koliko je blockchain tehnologija nova i koliko je nivo njene primene nizak. Inicijalna verzija Fabric-a je kreirana od strane IBM-a.

Implementacija Fabric-a je zasnovana na blockchain tehnologiji, ali pruža drugačije realizacije nekih koncepata. Pre svega, kod Fabric-a je veoma bitno da postoje različite vrste čvorova unutar mreže. Fabric uvodi koncept kanala koji omogućuju privatnost između nekih članova mreže, dok se pravila konsenzusa mogu implementirati na razne načine. Još jedna bitna karakteristika Fabric-a je uvođenje redundantne memorijske strukture. U nastavku poglavlja će biti opisani glavni činioci Fabric-a. Pored toga, na kraju poglavlja je dat opis toka transakcije, od njegog pozivanja do prihvatanja.

3.2.1. Vrste čvorova

U Hyperledger Fabric-a postoji jasna podela čvorova prema ulozi. Prema [7] postoje 3 grupe čvorova, a to su uređivači (engl. *orderers*), ravnopravni članovi (engl. *peers*) i klijenti. Na taj način se specifični poslovi bitni za funkcionisanje mreže delegiraju određenoj grupi čvorova.

3.2.1.1. Klijenti

Grupa klijenti predstavlja krajnje korisnike koji pozivaju transakcije. Klijenti komuniciraju sa mrežom preko aplikacionog programskog interfejsa (engl. *Application Programming Interface - API*) . Ova grupa čvorova komunicira i sa uređivačima i sa ravnopravnim članovima.

3.2.1.2. Ravnopravni članovi

Ravnopravni članovi (eng. *peers*) prestavljaju čvorove koji izvršavaju transakcije i održavaju dnevnik transakcija. Ravnopravni čvor može biti prihvatač (engl. *endorser*) ili potvrđivač (engl. *commiter*). Svaki ravnopravni član je potvrđivač, ali ne mora biti prihvatač. Potvrđivači dobijaju validirane transakcije u obliku blokova od strane uređivača i upisuju ih u dnevnik transakcija.

Prihvatač je čvor koji proverava validnost transakcije, na osnovu unapred definisanih pravila. Do njega stižu predlozi transkacija (engl. *transaction proposal*), a on proverava validnost transakcije i simulira njeno izvršenje kako bi se garantovala njena ispravnost. Prihvatači daju saglasnost za transakcije.

3.2.1.3. Uređivači

Uređivači su čvorovi koji imaju ulogu pakovanja transakcija odgovarajućim redosledom u blokove. Oni dobijaju transakcije, a njihovo uređivanje se radi prema postavljenim pravilima unutar mreže. Uređivački čvor predstavlja deo uređivačkog sloja (engl. *ordering service*). Ovaj sloj može biti organizovan kao centralizovan ili decentralizovan servis.

3.2.2. Smeštanje podataka

Fabric koristi dve komponente za smeštanje podataka. Sadrži dnevnik transakcija baš kao i ostale blockchain implementacije, ali postoji i baza stanja (engl. *world state*).

3.2.2.1. Baza stanja

Komponenta stanja odnosno baza stanja sadrži aktuelne vrednosti unutar blockchain-a. Baza stanja je organizovana kao kolekcija parova ključ-vrednost (engl. *key-value*). Ključ je oblika niza karaktera koji predstavljaju naziv, a vrednost je tipa BLOB. Baza stanja je redundantna memorijska struktura. Ona predstavlja indeksirani pogled na dnevnik transakcija i prema tome može se rekonstruisati iz njega. Svaka promena baze stanja se beleži. Prema [7] baza stanja može biti LevelDB ili CouchDB. Komponenta stanja je implementirana u cilju boljih performansi izvršavanja pametnih ugovora, odnosno provere stanja su omogućene bez prolaska kroz sve transakcije. Baza stanja podržava kompleksne upite u jeziku sličnom SQL-u .

3.2.2.2. Dnevnik transakcija

Dnevnik transakcija predstavlja izvor podataka za sve promene stanja (validne transakcije), ali i neuspešne pokušaje da se promeni baza stanja (nevalidne transakcije). Ova komponenta je organizovana slično kao i kod drugih blockchain implementacija, odnosno predstavlja potpuno uređenu strukturu blokova koji sadrže transakcije. Karakteristično za Fabric je da se i nevalidne transakcije smeštaju u blokove. Transakcije su unutar bloka potpuno uređene. Pošto postoji tačan redosled blokova i transakcija unutar blokova, to znači da je poznat i tačan redosled svih transakcija koje su se ikada desile u sistemu. Dnevnik transakcija se nalazi kod svih ravnopravnih članova koji imaju prava da mu pristupe, a može se naći i kod odgovarajućih uređivača.

U cilju brzog pristupa samo validnim transakcijama, pored stanja i dnevnika transakcija, ravnopravni članovi održavaju i dnevnik validnih transakcija (engl. *validated ledger*). Ovakva struktura predstavlja lanac blokova koji sadrže samo validne transakcije (engl. *vBlock*). Sadržaj ovakvih, validnih blokova je dobijen filtriranjem blokova koji sadrže validne i nevalidne transakcije. Svaki validini blok sadrži heš prethodnog validnog bloka, broj trenutnog validnog bloka, listu validnih transakcija koje pripadaju bloku iz kojeg je nastao validni blok i heš bloka od kog je validni blok nastao.

Veličina bloka u dnevniku transakcija može se dinamički podešavati unutar nekog kanala. Frekvenciju kreiranja blokova moguće je podešavati na osnovu broja transakcija ili vremenskog intervala.

3.2.3. Kanali

Kanali, koji su detaljno opisani u [7], se mogu posmatrati kao privatne podmreže za komunikaciju dva ili više učesnika blockchain mreže. Svaki kanal ima svoj dnevnik transakcija i bazu stanja. Samo čvorovi koji su pretplaćeni na kanal imaju uvid u transakcije i podatke koji se šalju kroz kanal. Svaki ravnopravni član koji je član kanala, čuva sopstvenu kopiju dnevnika transakcija i baze stanja za taj kanal. U dnevniku transakcija unutar kanala, prvi blok sadrži konfiguracione informacije. Član mreže može biti pretplaćen na više kanala.

3.2.4. Kod lanca

Pametni ugovori se u Fabric-u implementiraju programima koji se nazivaju kodovi lanca (engl. *chaincodes*). Njihova svrha je da implementiraju poslovnu logiku unutar nekog kanala ili čitave mreže. Trenutno se ovi programi pišu u jeziku Go, ali prema [7] će uskoro biti moguće pisati ih i u drugim programskim jezicima. Promena baze stanja se dešava pokretanjem koda lanca. Pored toga, ovakav program može da šalje upite nad bazom stanja i da eventualno menja bazu na osnovu dobijenih rezultata. Ulazna tačka za svaki chaincode je funkcija *main*, koja se pokreće kada ravnopravni član razvije (engl. *deploy*) svoju instancu koda lanca. Svaki ovakav kod implementira Chaincode intrefejs, odnosno mora da definiše metode *Init*, *Query* i *Invoke*.

Init metoda služi da inicijalizuje početna podešavanja neophodna za izvršavanje koda lanca. Poziva se kada se kod lanca prvi put postavi na blockchain.

Query metoda služi za slanje upita nad podacima. Ona ne može da promeni stanje dnevnika transakcija, odnosno baze stanja, pa se njeni pozivi ne beleže. Primer *Query* metode i pomoćne *read* metode su dati na slici 8. Poziv metode *stub.GetState(key)* pristupa bazi stanja i dobavlja vrednost koja odgovara ključu *key*.

```
func (t *ExampleChaincode) Query(stub shim.ChaincodeStubInterface,
function string, args []string) ([]byte, error) {
    // funkcija read mora biti definisana u kodu
    if function == "read" {
        return t.read(stub, args) //poziv read funkcije
    }
    fmt.Println("Nije pronadjena funkcija: " + function)

    return nil, errors.New("Dobijena nepoznata funkcija: " + function)
}

func (t *ExampleChaincode) read(stub shim.ChaincodeStubInterface,
args []string) ([]byte, error) {
  var key, jsonResp string
  var err error
    if len(args) != 1 {
        return nil, errors.New("Pogresan broj argumenata")
    }
    key = args[0] // kljuc cija se korespodentna vrednost trazi
    valAsbytes, err := stub.GetState(key)
    if err != nil {
        //json odgovor
        jsonResp = "{\"Greska\":\"Neuspesno dobavljanje stanja za " + key + "\"}"
        return nil, errors.New(jsonResp)
    }
    return valAsbytes, nil
}
```

Slika 8. Primer *Query* i pomoćne *read* metode

Invoke metoda se poziva da bi se promenili podaci unutar blockchain-a. Svaki poziv ove metode predstavlja transakciju koja se kasnije smešta u blok, odnosno u dnevnik transakcija. Primer *Invoke* metode je dat na slici 9, zajedno sa pratećom metodom *write*. Funkcija *stub.PutState(key,value)* se koristi za upis promenljive *key* u bazu stanja sa vrednošću *value*.

```
func (t *ExampleChaincode) Invoke(stub shim.ChaincodeStubInterface,
function string, args []string) ([]byte, error) {
    // obradjivanje razlicitih poziva funkcije u
    // zavisnosti od naziva funkcije koji je prosledjen kao argument
    if function == "otherFunctionName" {
        return t.Init(stub, "init", args)
    } else if function == "write" {
        return t.write(stub, args)
    }
    return nil, errors.New("Prosledjena nepoznata funkcija: " + function)
}

func (t *ExampleChaincode) write(stub shim.ChaincodeStubInterface,
args []string) ([]byte, error) {
  var key, value string
  var err error
  if len(args) != 2 {
    return nil, errors.New("Ocekivana dva argumenta, kljuc i vrednost")
  }
  key = args[0] //prvi argument je kljuc
  value = args[1] //drugi argument je vrednost
  err = stub.PutState(key, []byte(value))  //upis promenljive
  if err != nil {
    return nil, err
  }
  return nil, nil
}
```

Slika 9. Primer *Invoke* i pomoćne metode *write*

3.2.5. Digitalna dobra

Digitalna dobra (engl. *assets*) su digitalne predstave materijalnih ili nematerijalnih svojina. Digitalna dobra se u Fabric-u čuvaju kao kolekcija parova ključ-vrednost (u bazi stanja), gde je ključ naziv dobra, a vrednost može biti podatak koji opisuje stanje, količinu ili neki drugi tip informacije koji je pridružen tom dobru. Digitalna dobra se menjaju kodom lanca. Svaka promena stanja nekog dobra se beleži u dnevniku transakcija odgovarajućeg kanala.

3.2.6. Konsenzus

Hyperledger Fabric do konsezusa dolazi na komplikovaniji način nego što je to slučaj kod javnih mreža. Potrebno je osigurati da se dodavanje transakcija u dnevnik dešava samo ako su transakcije odobrene od strane odgovarajućih učesnika u mreži. Prema [7] razlikuju se tri faze u procesu sticanja konsenzusa u Fabric-u, a to su saglasnost, uređivanje i validacija:

- Saglasnost (engl. *endorsement*) – provera ispravnosti transakcije se vrši prema definisanim pravilima saglasnosti (engl. *endorsement policy*). Pravila saglasnosti mogu biti npr. da se zahteva da bar *k* od *n* učesnika potvrdi ili da baš određeni učesnici moraju da potvrde transakciju.

- Uređivanje (engl. *ordering*) – vrše se provere nad transakcijama i utvrđuje se njihov redosled u bloku.

- Validacija – vrše se dodatne provere, uključujući proveru zadovoljenja pravila saglasnosti (engl. *endorsement policy*) i dvostruke potrošnje[5] (engl. *double-spending*).

Faze konsenzusa i njihove uloge će detaljnije biti opisane u delu koji se tiče toka transakcije 3.2.8.

3.2.7. Pružalac servisa članstva

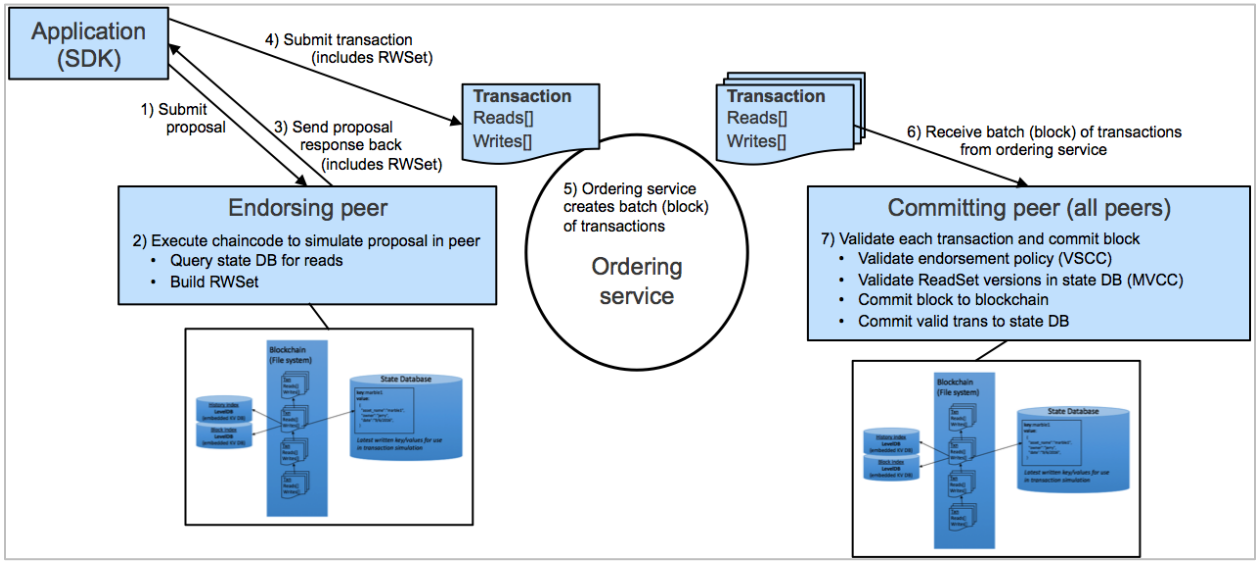
Pružalac servisa članstva (engl. *Membership Services Provider – MSP*) je komponenta koja ima za cilj da apstrahuje upravljanje članstvom mreže u Fabric-u. MSP sakriva sve mehanizme i protokole iza izdavanja i potvrđivanja sertifikata i identifikacije korisnika [7].

Hyperledger Fabric mrežom može da upravlja jedan ili više MSP-ova. Ovo obezbeđuje modularnost operacija i podršku za razne standarde članstva i arhitekture. U prethodnim verzijama Fabric-a, bio je samo jedan MSP zadužen za rad sa mrežom, pa je to potencijalno bila jedna tačka otkaza (SPOF). Komponenta Fabric Certificate Authority (CA), na koju se pružalac članstva oslanja je zadužena za registraciju identiteta, izdavanje sertifikata, oduzimanje i obnavljanje sertifikata.

[5] Problem trošenja istih sredstava više puta

3.2.8. Tok transakcije

Hyperledger Fabric upravlja transakcijom na komplikovaniji način nego što je to slučaj kod Bitcoin-a. Tek kroz opis životnog ciklusa transakcije sve komponente Fabric-a i njihove uloge postaju jasne. Na slici 10. je prikazan tok transakcije u Fabric-u [30].



Slika 10. Tok transakcije u Hyperledger Fabric-u

Pretpostavka za tok na slici je postojanje aktivnog kanala u čiji će se dnevnik transakcija beležiti novopristigle transakcije. Podrazumeva se da je korisnik aplikacije registrovan i da mu je dodeljen odgovarajući sertifikat i kriptografski podaci kako bi mogao pristupiti mreži odnosno kanalu. Takođe, postoji kod lanca, sa definisam pravilima saglasnosti, koji je instaliran na čvorovima (ravnopravnim članovima) i instanciran je unutar kanala.

Prvi korak je da aplikacija pošalje predlog transakcije do prihvatalačkih članova (koji daju saglasnost). Zatim, u drugom koraku prihvatalački član izvrši transakciju nad trenutnim podacima u bazi stanja. U ovom koraku baza stanja se ne menja. Nakon toga, ovaj član prikuplja skup čitanja i pisanja (engl. *read-write set*). Skup čitanja (engl. *read set*) sadrži listu jedinstvenih ključeva iz baze stanja i verzije podataka. Verzija podataka se koristi u poslednjiem koraku prihvatanja transakcije. Skup pisanja (engl. *write set*) sadrži listu jedinstvenih ključeva i nove vrednosti koje odgovaraju tim ključevima. Takođe, ukoliko se neki podatak briše umesto vrednosti za odgovarajući ključ se postavlja znak za brisanje (engl. *delete marker*).

U 3. koraku se šalje odgovor do klijentske aplikacije. Odgovor sadrži kriptografski materijal predloga transakcije i potpis člana koji šalje odgovor kao i skup čitanja i pisanja. U 4. koraku, klijentska aplikacija prikuplja sve odgovore od prihvatalačkih članova i pakuje ih u jedan kriptografski potpisan zahtev i zatim šalje do uređivačkog sloja. Uređivački sloj proverava sav kriptografski materijal dobijen od aplikacije. Nakon toga, proverava se da li postoji saglasnost svih relevantnih članova kanala da se kod lanca izvrši. Ta pravila mogu zahtevati da svi članovi unutar kanala moraju da daju saglasnost ili da nekakav drugačiji logički uslov bude ispunjen. Ako uređivački članovi ne dobiju saglasnost od svih potrebnih prihvatilačkih članova, transakcija će biti sačuvana u dnevniku transakcija, ali kao nevalidna. Ovakva transakcija neće promeniti bazu stanja. Čuvanje nevalidnih transakcija omogućuje postojanje istorijskih podataka o akcijama članova mreže ili kanala pa se može ustanoviti razlog nastanka problema ili propusta. Nakon ove provere uređivački sloj mora da uporedi sve dobijene skupove čitanja i pisanja dobijene od prihvatalačkih članova. Ukoliko nisu svi dobijeni skupovi čitanja i pisanja identični, transakcija će biti označena kao nevalidna, ali će se upisati u dnevnik transakcija. Peti korak podrazumeva uređivanje transakcija unutar bloka. U 6. koraku se šalje potpisani blok do svih ravnopravnih članova. Poslednji korak obuhvata dodatne provere kao što je test duplog trošenja, tako što se ispita da li je u periodu između simulacije transakcije i konačne potvrde transakcije promenjena verzija nekog podatka u bazi stanja. Svaki predlog transakcije menja verzije podataka u bazi stanja. Nakon uspešnog prolaska testova, dolazi do upisa bloka u dnevnik transakcija i posledično se menja baza stanja. Preostalo je da se aktivira događaj (engl. *event*) koji će obavestiti klijentsku aplikaciju o uspešnosti izvršenja predložene transakcije. Detaljniji opis koraka prihvatanja transakcije je dostupan u [7].

3.3. Poređenje Hyperledger Fabric-a i Bitcoin-a

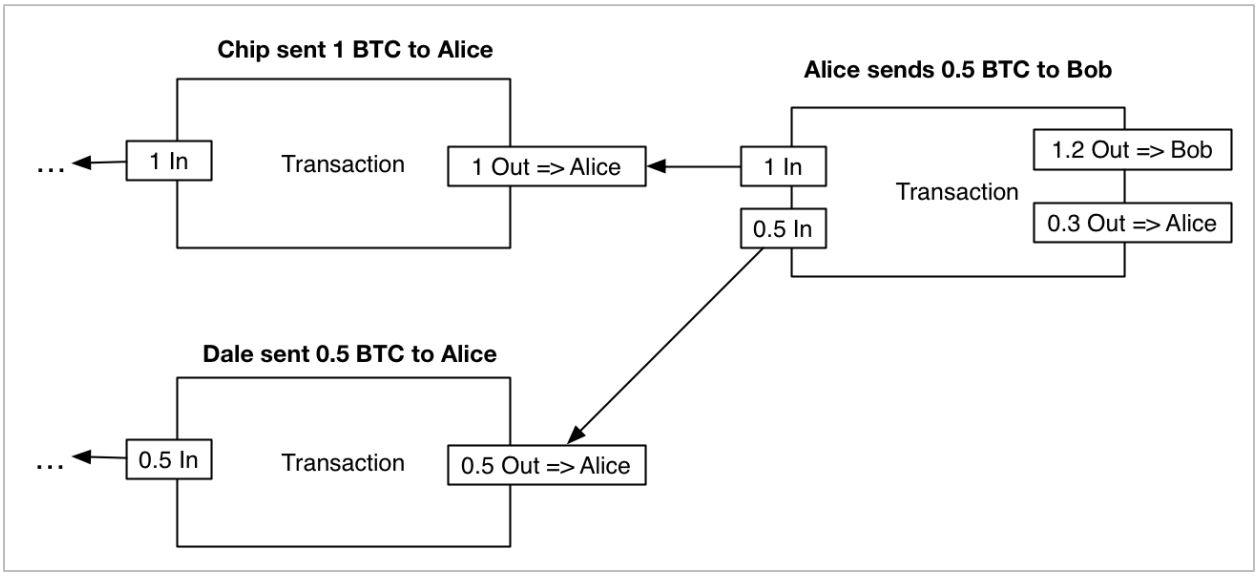
Bitcoin je najpoznatija kriptovaluta današnjice. Ova prva implementacija blockchain-a je i najzaslužnija za popularizaciju cele tehnologije. Bitcoin je prvi put opisan u [1], 2008. godine. Svi koncepti koji su opisani u drugom poglavlju važe i kod Bitcoin-a, samo se implementacioni detalji mogu razlikovati.

3.3.1. Osvrt na Bitcoin

Bitcoin mrežu ne uređuje nikakav sertifikovani centar, članovi mreže sami održavaju mrežu ispravnom [1]. Ne postoji hijererhija među članovima, svi su ravnopravni. Način transfera Bitcoin-a je sličan kao i transfer drugih valuta kao što su evro ili dinar. Svaki član mreže ima svoju adresu, koja predstavlja ekvivalent broju bankovnog računa. Preciznije je reći da svaka

klijentska aplikacija ima svoju adresu, jer korisnik može imati instaliranih više Bitcoin klijentskih aplikacija.

Korisnici su anonimni, njihov identifikator je negovoreća adresa. Bitcoin adresa sa pratećim podacima, privatnim i javnim ključevima se čuva u klijentskoj aplikaciji u digitalnom novčaniku (engl. *wallet*). Kroz klijentske Bitcoin aplikacije, korišćenjem podataka iz digitalnog novčanika, vrše se transakcije između korisnika. Kod Bitcoin-a, transakcija predstavlja prenos neke količine sredstava sa jedne Bitcoin adrese na drugu (koja ne mora nužno biti različita). Svaka transakcija ima ulaze i izlaze. Izlaz jedne transakcije predstavlja ulaz u drugu (Slika 11) [29]. Kusur se vraća tako što se kreira transakcija koja šalje potrebnu količinu sredstava sa jedne adrese na tu istu adresu. Provizija od transakcija je razlika u vrednosti ulaza i izlaza. Proviziju dobijaju čvorovi koji ulančaju novi blok u lanac. Veća provizija posledično znači bržu potvrdu transakcije [20].



Slika 11. Ulazi i izlaz u Bitcoin transakciji

Valuta Bitcoin nema ni fizičku ni digitalnu reprezentaciju. Dokaz da neko poseduje neku količinu kriptovalue su zapravo transakcije koje su upisane u dnevnik transakcija i čija rezultanta predstavlja stanje računa koje je povezano sa adresom digitalnog novčanika.

Komunikacija u Bitcoin-u se odvija kroz *peer-to-peer* mrežu [1]. Klijentska aplikacija pripremljenu transakciju šalje do svih ravnopravnih čvorova (članova) sa kojima je povezana, zatim svaki od tih čvorova šalje svojim povezanim čvorovima i na taj način se do svih članova mreže propagira transakcija. Čvorovi proveravaju ispravnost transakcija, a rudari ispravne transakcije smeštaju u blokove i pružaju dokaz posla. Nakon uspešnog pronalaska dokaza posla oni novi blok smeštaju u dnevnik transakcija i šalju svojim povezanim čvorovima novi blok. Drugi čvorovi koji dobiju blok proveravaju sve transakcije unutar bloka. Osim provera transakcija oni takođe proveravaju da li heš bloka ispunjava uslove (da li ima manju vrednost od

ciljne) i da li *nonce* odgovara hešu. Ukoliko je blok prošao testove, upusuje se u dnevnik transakcija čvora i propagira svim povezanim čvorovima koji rade isti postupak. Transakcija se tek smeštanjem bloka kom pripada u dnevnik transakcija proglašava potvrđenom, do tada se smatra nepotvređnom (nevažećom). Za veće transakcije se može zahtevati da se ulanča još šest blokova posle bloka kom transakcija pripada da bi se ona smatrala važećom. Smatra se da je nakon šest blokova verovatnoća da se blok unutar lanca izmeni minimalna.

U odnosu na bankarske sisteme, transferi su kod Bitcoin-a mnogo brži i koštaju mnogo manje. Svaka transakcija je potpuno transparentna, može se videti sa koje adrese je prebačeno koliko sredstava, ali se ne zna kome pripadaju te adrese, tako da postoji potpuna privatnost. Podaci koje banka čuva lakše mogu biti kompromitovani ili izgubljeni, jer svaki član Bitcoin mreže može čuvati svoju kopiju svih transakcija koje su se ikada desile unutar Bitcoin mreže.

Nije poznato ko se krije iza pseudonima Satoši Nakamoto [26]. On je implementirao prvu verziju Bitcoina u januaru 2009. godine. Takođe, on je izrudario početni blok (engl. *genesis block*) koji ne referencira prethodni i zbog toga se razlikuje od svih narednih blokova. Prvu transakciju ikada načinio je Nakamoto poslavši 10 BTC-a Hal Finney-u [27]. Najčuvenija transakcija desila se u maju 2010. godine kada je Laszlo Hanyecz potrošio 10000 BTC-a za dostavljanje 2 pice [27]. Zanimljivo je da 13. septembra 2017. godine, 10000 Bitcoina vredi oko 39 000 000 dolara. Satoši Nakamoto se potpuno povukao iz razvoja projekta 2011. godine i prepustio je dalji razvoj Gavinu Andersonu. Prema [27], pretpostavlja se da je u ranom periodu Bitcoin-a Nakamoto izrudario i sada poseduje oko 1 000 000 Bitcoin-a, što 13. septembra 2017. godine vredi približno 3 910 000 000 dolara. Na slici 12. prikazana je promena vrednosti Bitcoina u periodu od juna 2013. do septembra 2017. godine [21].

Slika 12. Promena vrednosti Bitcoin-a sa vremenom

3.3.2. Razlike Hyperledger Fabric-a i Bitcoin-a

Hyperledger Fabric i Bitcoin funkcionišu na platformi blockchain-a, ali postoje mnogobrojne razlike između ove dve implementacije. Pre svega Hyperledger Fabric i Bitcoin se razlikuju po nameni. Bitcoin mreža služi za razmenu kriptovalute. Fabric, sa druge strane, ima mnogo širi opseg primene, on ima za cilj da podrži razne zahteve poslovanja. Dakle, predmet transakcija kod Bitcoin-a je uvek Bitcoin valuta, dok su kod Fabric-a to digitalna dobra, koja predstavljaju materijalne i nematerijalne svojine. Pošto je Bitcoin mreža javna, a Fabric privatna, postoje razlike u masovnosti mreže, načinu dostizanja konsenzusa i u vrsti čvorova. U [28], pretpostavlja se da Bitcoin mreža u 2017. godini ima između 5.8 i 11.5 miliona aktivnih digitalnih novčanika. Bitcoin mreža je jedna, globalna, dok Hyperledger Fabric-a mreža može biti mnogo. Kod Bitcoin-a korisnici su anonimni ali ih ima mnogo, pa je pogodan način postizanja konsenzusa kroz dokaz posla. Kod Fabric-a korisnici su identifikovani i moguće je primeniti drugačije algoritme kao što je Kafka [7], kako bi se prihvatile ispravne a odbacile neispravne transakcije. Algoritmi za dostizanje konsenzusa kod Fabric-a mogu jednostavno biti promenjeni u zavisnosti od potrebe mreže. U tabeli 2. prikazane su razlike između Bitcoin-a i Fabric-a.

	Hyperledger Fabric	Bitcoin
Opis platforme	Blockchain opšte namene	Blockchain za transfer kriptovalute
Razvoj	Linux Foundation	Programeri Bitcoin-a
Valuta	/	Bitcoin (BTC)
Nagrade za rudarenje	/ (Nema rudarenja)	Bitcoin
Skladište podataka	Baza stanja i dnevnik transakcija	Dnevnik transakcija
Konsenzus	(Kafka, PBFT) Promenljiv po potrebi	Dokaz posla
Otvorenost mreže	Privatna	Javna
Privatnost	Korišćenjem kanala	/
Pametni ugovori	Da, jezik Go	Rudimentarni, skript jezik
Identitet korisnika	Poznat	Anoniman

Tabela 2. Razlike Hyperledger-a i Bitcoin-a

Bitcoin dozvoljava transparentan prikaz svih transakcija koje su se desile. Svima su dostupni svi podaci i ne postoji način da se obezbedi privatnost u komunikaciji između grupe čvorova. U Fabric-u verzije 1.0, koncept kanala je jedan od najbitnijih, jer omogućuje da grupa čvorova komunicira potpuno privatno. Svaki kanal ima sopstvene podatke koji nisu dostupni čvorovima koji nisu eksplicitno pretplaćeni na kanal.

Fabric je mnogo brži u potvrđivanju transakcija, samim tim što nije potrebno rešavati vremenski zahtevan matematički problem da bi se potvrdio blok. Kod Fabric-a podaci se čuvaju

u 2 memorijske komponente, u dnevniku transakcija i bazi stanja. Bitcoin sa druge strane se oslanja samo na dnevnik transakcija.

U slučaju Fabric-a, kod lanca je jedna od osnovnih komponenata i predstavlja implementaciju ugovora. Kod lanca se implementira u jeziku Go, a uskoro će to biti moguće uraditi i u nekim drugim programskim jezicima [7]. Bitcoin nema razvijene pametne ugovore. Koristi skript jezik koji produkuje kod koji sadrži listu instrukcija koje opisuju šta primalac sredstava iz transakcije mora da obezbedi da bi koristio namenjena sredstva [10]. Podrazumevano, traži se da poseduje javni ključ koji heširanjem daje odredišnu adresu iz transakcije. Pored toga, potrebno je da primalac sredstava obezbedi dokaz o posedovanju privatnog ključa koji odgovara javnom ključu.

4. Primena blockchain distribuiranih sistema u oblasti farmacije

U procesu dobavljanja lekova postoji mnogo aktera, počev od proizvođača do krajnih korisnika. Svako od njih bi trebalo da se najstrožije pridržava pravila, jer u suprotnom moglo bi doći do posledica po ljudsko zdravlje. Po [16], 10% svih lekova na tržištu je lažno, dok se prema [15] oko 50% svih lekova kupljenjih preko internet smatra ilegalnim. Neki od ilegalnih lekova nemaju efekat kakav bi trebalo i ne doprinose poboljšanju zdravstvenog stanja, dok neki imaju u sebi otrovne supstance. Sastojci lekova se često nabavljaju iz više zemalja, pakovanje se takođe može dešavati u drugoj zemlji i transportovati na razne načine od strane više kompanija. Ovakav sistem rada ostavlja mnogo prostora za ilegalne radnje.

Sa druge strane, veleprodajne i maloprodajne kompanije imaju česta neslaganja koja se tiču međusobnog poslovanja jer svaka učesnik ima sopstvene podatke. Velike kompanije koje posluju sa hiljadama partnera, moraju da uskladi svoje podatke sa svakim od njih, a taj proces je vremenski veoma zahtevan.

Blockchain sistemi bi mogli da reše ili umanje pomenute probleme. Svaka transakcija bi morala biti odobrena od strane relevantnih aktera unutar mreže koji bi imali uvid u transakcije, a poslovne obaveze bi se automatski izvršavale primenom pametnih ugovora. Za realizaciju idejnog rešenja u vezi nabavke farmaceutskih proizvoda korišćen je alat Fabric Composer [4].

4.1. Fabric Composer

Fabric composer je radni okvir (engl. *framework*) koji pojednostavljuje razvoj i testiranje Hyperledger Fabric aplikacija. On omogućava da se razvijaju aplikacije bez detaljnog znanja o načinu funkcionisanja blockchain tehnologija. Composer je fokusiran na implementaciju poslovne logike i pruža jednostavan način da se modeluju učesnici unutar mreže i transakcije među njima. Dodatni alati omogućuju da se za aplikacije kreirane Composer-om generišu REST API kao i frontend aplikacija uz pomoć Yeoman-a. Composer ima svoje igralište (engl. *playground*) koje pruža jednostavan grafički interfejs za definisanje i testiranje poslovne mreže. Postoji mogućnost korišćenja ovog alata lokalno ili kroz *web* pretraživač.

Definicija poslovne mreže se nalazi u nekoliko fajlova. Modelovanje digitalnih dobara koja su opisana u 3.2.5. i učesnika (engl. *participant*) u mreži se vrši u model fajlu (.cto). Za ovu svrhu se koristi poseban jezik za modelovanje. Svaki model fajl pripada nekom imenskom prostoru (engl. *namespa*ce). Moguće je uključiti definicije iz drugih imenskih prostora korišćenjem ključne reči *import.* Kod koji opisuje jedan model digitalnog dobra, liči na kod koji bi se pisao u npr. Javi za modelovanje tipa entiteta iz relacione paradigme. U ovom jeziku za modelovanje postoje prosti tipovi podataka (String, Double, Integer, DateTime, Boolean i Long), a postoje i složeni tipovi podataka. Takođe, postoje i nizovi prostih ili složenih tipova. Atributima koji su prostog tipa mogu biti dodeljene i podrazumevane vrednosti, a mogu se pisati

i regularni izrazi za njihovo validiranje. Svako digitalno dobro ili učesnik sadrži jedinstveni identifikator.

Postoje apstraktni tipovi kao i koncept nasleđivanja koje se često koristiti kada se modeluju učesnici u mreži. Pored pomenutog, enumeracija se takođe može definisati u .cto fajlu. U ovom fajlu (.cto) se takođe modeluju i transakcije, odnosno piše se njihova deklaracija, dok se njihova razrada piše u posebnom JavaScript fajlu. Pored pomenutih komponenti ovog jezika postoje i koncepti (engl. *concepts*) koji prestavljaju složene tipove koji nisu transakcije, učesnici ili digitalna dobra. Najčešće koncepti budu referencirani u definiciji učesnika ili digitalnog dobra.

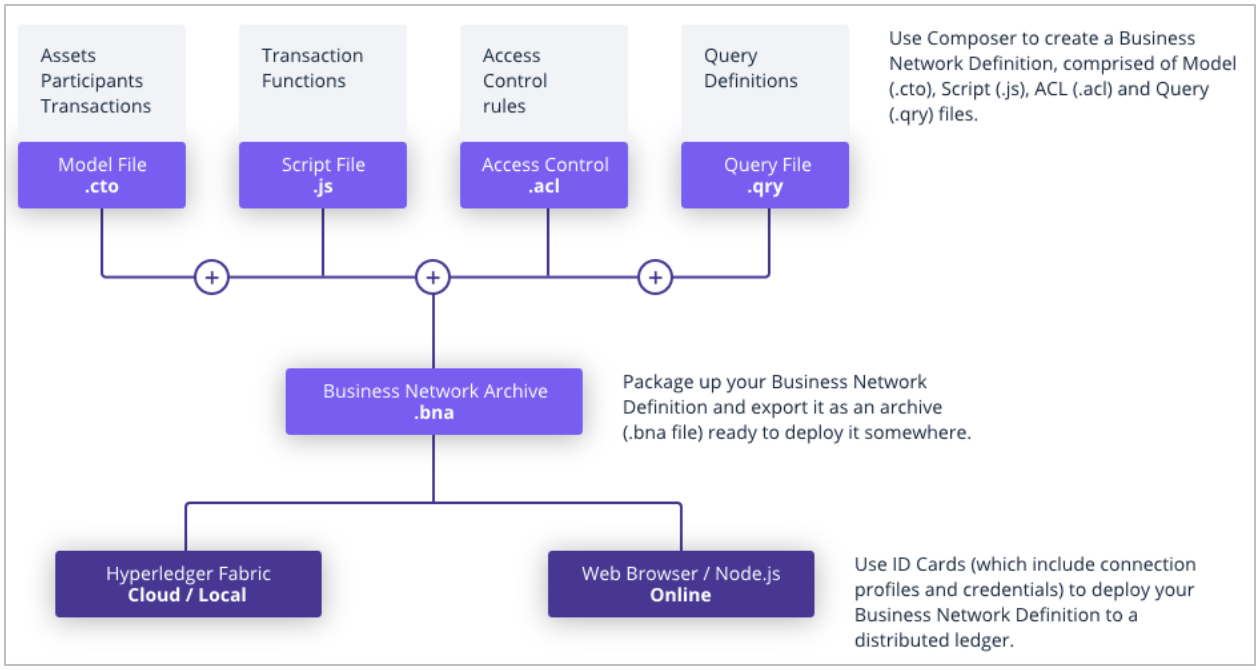
Implementacija transakcija, odnosno poslovne logike i drugih pomoćnih funkcija se piše odvojenom fajlu. Razrada transakcije mora da odgovara deklaraciji transakcije iz .cto fajla. Kod Composer-a definicija poslovne logike se piše u JavaScript-u koji postaje sve popularniji i na taj način omogućuje velikom broju ljudi da započnu korišćenje Composer-a. Nasuprot tome, ukoliko se koristi Fabric direktno, potrebno je znati Go jezik da bi se pisali pametni ugovori.

Vrlo bitno za Fabric je definicija prava koje učesnici imaju nad određenim digitalnim dobrima. Kod Composer-a ovakva prava pristupa se definišu unutar posebnog .acl **(**Access Control Lists) fajla. U ovim fajlovima se mogu pisati i kompleksni uslovi na osnovu kojih se proverava da li konkretni učesnik sme da čita ili piše unutar nekog dnevnika digitalnih dobara [6](engl. *asset registry*). Definicijom pravila unutar .acl fajlova se može postići da npr. samo član koji je kreirao neku narudžbenicu ima pravo da je menja i to najviše sat vremena od trenutka njenog kreiranja.

Postoje i .qry fajlovi koji dozvoljavaju pisanje kompleksnih i parametrizovanih upita nad podacima. Jezik koji se koristi liči na SQL. Definisani upiti se pozivaju iz koda unutar transakcija. Postoji i mogućnost dinamičkog kreiranja i pozivanja upita.

Svi pomenuti fajlovi zajedno čine definiciju poslovne mreže. Arhiva mreže .bna fajl se zatim spušta na Hyperledger Fabric platformu (Slika 13) [4].

[6] Perzistentna kolekcija digitalnih dobara u Composer-u, za svaki tip dobra postoji posebna kolekcija. Analogne strukture postoje i za učesnike.



Slika 13. Arhiva poslovne mreže u Fabric Composer-u

Da bi učesnik mogao da komunicira sa ostatkom mreže, on mora imati neki identifikacioni dokument. Potrevno je kreirati novu instancu učesnika odgovarajućeg tipa. Nakon toga se mora izdati identifikacioni dokument i dodeliti novokreiranoj instanci. Korisnik zatim može koristiti identifikacione podatke iz tog dokumenta kako bi koristio mrežu u skladu sa pravima pristupa. Moguće je i da korisnik priloži neki identifikacioni dokument i da se samo kreirana instanca učesnika poveže sa priloženim dokumentom. Identifikacioni dokumenti mogu da isteknu, što će dovesti do njihovog povlačenja (engl. *revoke*) i onemogućtii učesnika, kom povučeni dokument pripada da komunicira sa ostatkom mreže.

4.2. Opis rešenja

U ovom delu će biti opisana definicija poslovne mreže koja bi trebalo da podrži proces nabavke u farmaceutskoj industriji. U 4.2.1. je dat kratak opis procesa dostave farmaceutskih proizvoda, dok je u 4.2.2. dat detaljan opis poslovne mreže i aplikacije koja se generiše na osnovu nje.

4.2.1. Opis procesa

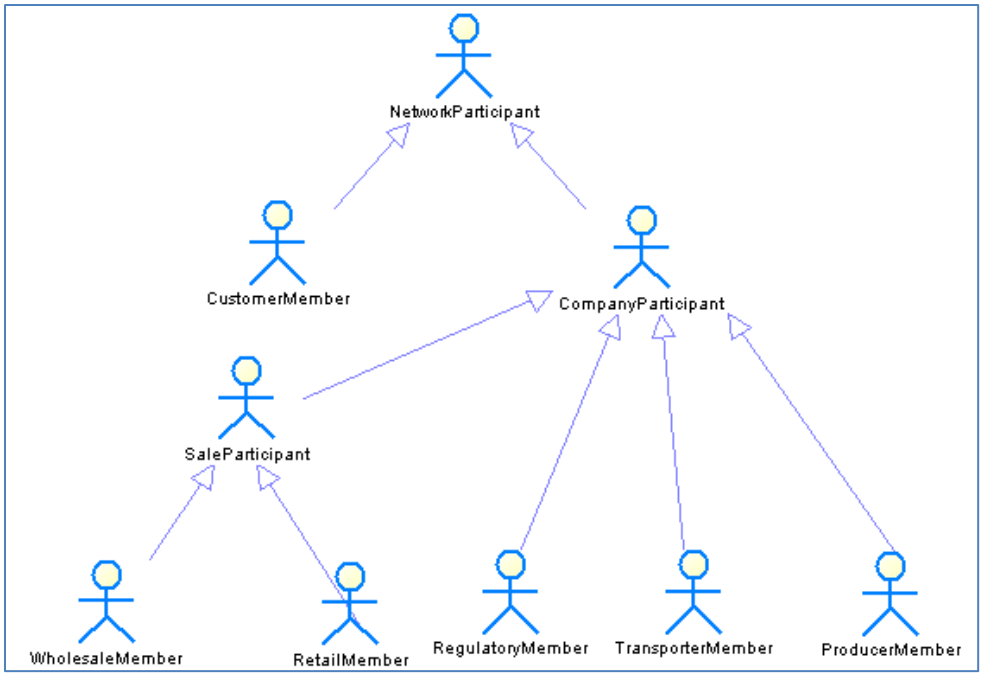
Proizvodi[7] se uvoze ili proizvode, a svaki lek mora dobiti atest od strane regulatornog tela. U slučaju uvoza, uvoznik predstavlja primarnog distributera, koji proizvode distribuira veleprodajama ili maloprodajama. Pored toga, veleprodajna kompanija može distribuirati proizvode i pravnim licima kao što je bolnica. Maloprodajna kompanija može da poručuje proizvode direktno od proizvođača, iako češće sarađuje sa veleprodajnim kompanijama. Transportom proizvoda bave se kompanije od kojih se roba dobavlja ili kurirske službe, ali tako da se lekovi prevoze u posebnim uslovima. Krajnji korisnici proizvode nabavljaju od maloprodajnih kompanija.

4.2.2. Opis implementacije

Kako je za realizaciju rešenja korišćen Fabric Composer, potrebno je opisati sve neophodne elemente koji čine definiciju poslovne mreže, a to su učesnici, digitalna dobra i transakcije. Pored toga biće potrebno da se specifikuju i .qry fajlovi koji će sadržati upite.

Na osnovu opisanog procesa identifikovani su učesnici: krajnji korisnici, maloprodajne, veleprodajne, transportne kompanije, proizvođači i kontrolni akteri (Agencija za lekove i medicisnka sredstva i Republički fond za zdravstveno osiguranje). U ovom rešenju neće se razmatrati bolnice, a podrazumevaće se da se uvoz vrši veleprodajna firma. Takođe, rešenje neće razmatrati lekari koji izdaju recepte pacijentima. Ovi učesnici su modelovani na osnovu procesa poslovanja u Srbiji, ali je proces sličan i u drugim državama. Hijerarhija učesnika je prikazana na slici 14.

[7] Farmaceutske kompanije ne prodaju samo lekove, iako oni čine većinu njihovog prodajnog asortimana.



Slika 14. Hijerarhija učesnika u sistemu

Učesnik u mreži je modelovan kao apstraktan tip *NetworkParticipant* koji se identifikuje preko adrese elektronske pošte. Pored *balance* atributa koji opisuje stanje na računu učesnika, postoje i dodatna obeležja koja bliže opisuju korisnika. Produkciono rešenje bi podrazumevalo dodatne atribute koji se tiču identifikacije korisnika. Modeli konkretnih grupa korisnika nasleđuju pomenuti apstraktni tip. Pravni subjekti se mogu modelovati kao jedan apstraktni tip (*CompanyParticipant)* Ovaj tip bi za produkciono rešenje podrazumevao obeležja kompanije kojoj učesnik pripada. Ti podaci bi uključivali detaljnje podatke o kompaniji, kao što su PIB, matični broj i šifra delatnosti. Maloprodajna (*RetailMember*) i veleprodajna grupa korisnika (*WholesaleMember*) nasleđuju aprstraktni tip SaleMember koji ima i listu dolazećih narudžbenica (*incomingOrders*) u cilju praćenja pristiglih porudžbina. Pored toga, oni poseduju i listu konkretnih proizvoda (*incomingProductInstances*) koji bi trebalo da pređu u njihov posed (naručeni proizvodi). Takođe, korisno je voditi evidenciju o količni proizvoda na stanju koji se čuvaju u kolekciji *amountAvailible*. Vrlo često dostavu proizvoda radi veleprodajna odnosno maloprodajna firma, bez posredstva transportne kompanije. Transportne kompanije, regulatorna tela i krajni korisnici su modelovani tipovima *TransportMember*, *RegulatoryMember* i *Customer*, respektivno. Model učesnika u sistemu je prikazan na slici 15.

```
abstract participant NetworkParticipant identified by participantEmail {
  o String participantEmail
  o String participantFirstName
  o String participantLastName
  o Double participantBalance
}

abstract participant CompanyParticipant extends NetworkParticipant {
  o String companyName
}

abstract participant SaleParticipant extends CompanyParticipant {
  --> Order [] incomingOrders optional
  --> ProductInstance [] incomingProductInstances optional
  --> Product [] amountAvailiable
}

participant WholesaleMember extends SaleParticipant {
}

participant RetailMember extends SaleParticipant {
}

participant TransportMember extends CompanyParticipant {
}

participant ProducerMember extends CompanyParticipant {
}

participant RegulatoryMember extends CompanyParticipant {
}

participant Customer extends NetworkParticipant {
}
```

Slika 15. Model učesnika u sistemu

Modeli regulatornih članova, proizvođača i krajnjih korisnika sadrže dodatne atribute koji nisu od interesa za ovo idejno rešenje, te neće biti razmatrani.

Kod relacionih baza podataka se svi korisnici smeštaju u istu relaciju, a uloga korisnika predstavlja obeležje koje implicira prava pristupa korisnika. Nasuprot tome, u Composer-u se pravila pristupa digitalnim dobrima definišu za svaku grupu korisnika posebno, pa se zato modelovanje učesnika vrši nasleđivanjem.

U Composer-u svaki konkretan tip učesnika ima posebnu perzistentnu kolekciju (engl. *participant registry*) u kom se čuvaju instance tog tipa. To implicira da ukoliko je unutar implementacije transakcija potrebno napraviti promenu nekog učesnika koji je apstraktan, mora se dinamički odrediti kog konkretnog tipa je učesnik. Isto važi i za digitalna dobra.

Što se tiče digitalnih dobara u sistemu identifikovani su: klase proizvoda, instance proizvoda i narudžbenice sa svojim stavkama. Model digitalnih dobara u Composer-u je dat na slici 16.

```
asset Product identified by productName {
  o String productName
  o String productBarcode
  o String productDescription
  o ProductType productType
  o Double productPrice optional
}
asset ProductInstance identified by productInstanceId {
  --> Product product
  --> NetworkParticipant owner
  o String productInstanceId
  o DateTime producedDateTime optional
  o Double productInstancePrice
  o ProductStatus status optional
}
asset OrderItem identified by orderItemId {
  --> Product product
  --> ProductInstance [] instances
  o String desc optional
  o String orderItemId
  o Double amount
}
asset Order identified by orderId{
  --> OrderItem[] items
  --> NetworkParticipant sender
  --> NetworkParticipant receiver
  o String orderId
  o Double totalPrice optional
  o DateTime createdTime optional
  o OrderStatus orderStatus default = "CREATED" optional
}
```

Slika 16. Model digitalnih dobara u Composer-u

Kako je vrlo često u interesu da se zna odakle je svako pakovanje proizvoda došlo, potrebno je da se nivo identifikacije spusti u odnosu na tradicionalne poslovne sisteme tako da se evidentira svako pakovanje proizvoda.

Svaka klasa proizvoda (*Product*) predstavlja klasu konkretnih proizvoda (pakovanja). Klasa proizvoda sadrži naziv, bar-kod, opis, osnovnu cenu, tip proizvoda koji govori o tome da li je proizvod lek ili ne. Pored toga klasa proizvoda bi u narednim implementacijama trebalo da sadrži i grupu proizvoda kojoj pripada (kozmetika, ortopedska pomagala, analgetici, vitamini, i drugi).

Konkretan proizvod referencira klasu proizvoda kojoj pripada i pripada akteru unutar mreže. Instanca proizvoda (*ProductInstance*), odnosno pakovanje ima status koji govori o tome da li je instanca naručena, odnosno da li je u fazi slanja. Svako pakovanje proizvoda može da ima redefinisanu cenu u odnosu na klasu proizvoda kojoj pripada, u slučaju da se npr. rok trajanja bliži isteku.

Order predstavlja model za dokument narudžbenicu i dokument koji je odgovor na nju. Tip *Order* referencira pošilaoca (*sender*) i primaoca (*receiver*) narudžbenice. Stavke narudžbenice su predstavljene kolekcijom *items*. Pored toga, ovaj tip sadrži i ukupnu cenu naručenih stavki (*totalPrice*), vreme kreiranja (*createdTime*) kao i status dokumenta (*orderStatus*).

Stavka dokumenta narudžbenice je modelovana tipom *OrderItem* koji sadrži referencu na klasu naručenog proizvod (*product*), kao i traženu količinu koja se odnosi na traženi proizvod (*amount*). Konkretne instance proizvoda koji će biti isporučeni kao odgovor na stavku narudžbenice su opisane kolekcijom *instances*.

U sistemu su takođe opisani tipovi nabrajanja (enumeracije). OrderStatus enumeracija služi da iskaže stanje narudžbenice. Moguća stanja su CREATED (kreirana), RECEIVED (dostavljena do primaoca), NOT_APPROVED (nije odobrena od strane primaoca), IN_PROGRESS (obrađuje se) SHIPPING (u fazi dostave), FINISHED (narudžbenica je obrađena, a roba je dostavljena poručiocu). ProductType može imati vrednost DRUGS koja govori da se radi o leku i NOT_DRUGS u slučaju da se radi o proizvodu koji nije lek. Potreba za distinkcijom između ova dva pojma proizilazi iz drugačijeg načina skladištenja i transporta. ProductStatus predstavlja tip nabrajanja koji se pridružuje instanci proizvoda da ukaže na njeno stanje. Ova enumeracija može imati vrednost DEFAULT (uobičajeno stanje), ORDERED (instanca je naručena, odnosno rezervisana) i SHIPPING (instanca je u fazi transporta).

Iz opisa procesa lanca dostavljanja se može zaključiti da je najvažnija operacija koju je potrebno modelovati, operacija promene vlasništva jednog proizvoda. Ovakva poslovna aktivnost se ne može preslikati na jednu nedeljivu operaciju, odnosno mora se proći niz operacija da bi se promenilo vlasništvo instance proizvoda. Sledi opis ovih operacija.

SendOrder transakcija podrazumeva slanje već kreirane narudžbenice akteru kojem je namenjena. Ako pošiljalac nije već slao tu narudžbenicu i ako ima dovoljno sredstava na računu, poslata narudžbenica se smešta u dolazeće narudžbenice primaoca. Ova operacija kao i sve ostale podrazumeva promenu stanja dokumenta.

ApproveAndPrepareOrder operacija proverava da li svaki traženi proizvod postoji u dovoljnoj meri kod primaoca narudžbenice. Ukoliko se ova provera prođe uspešno onda se svakoj stavki narudžbenice pridružuju instance proizvoda koje će biti dostavljene. Broj pridruženih instanci jednak je traženoj količini klase proizvoda kojoj pripada instanca. Prilikom slanja narudžbine pošiljalac specificira klasu proizvoda koje poručuju, a konkretne kutije koje će biti dostavljene su odabrane na slučajan način od strane primaoca narudžbine.

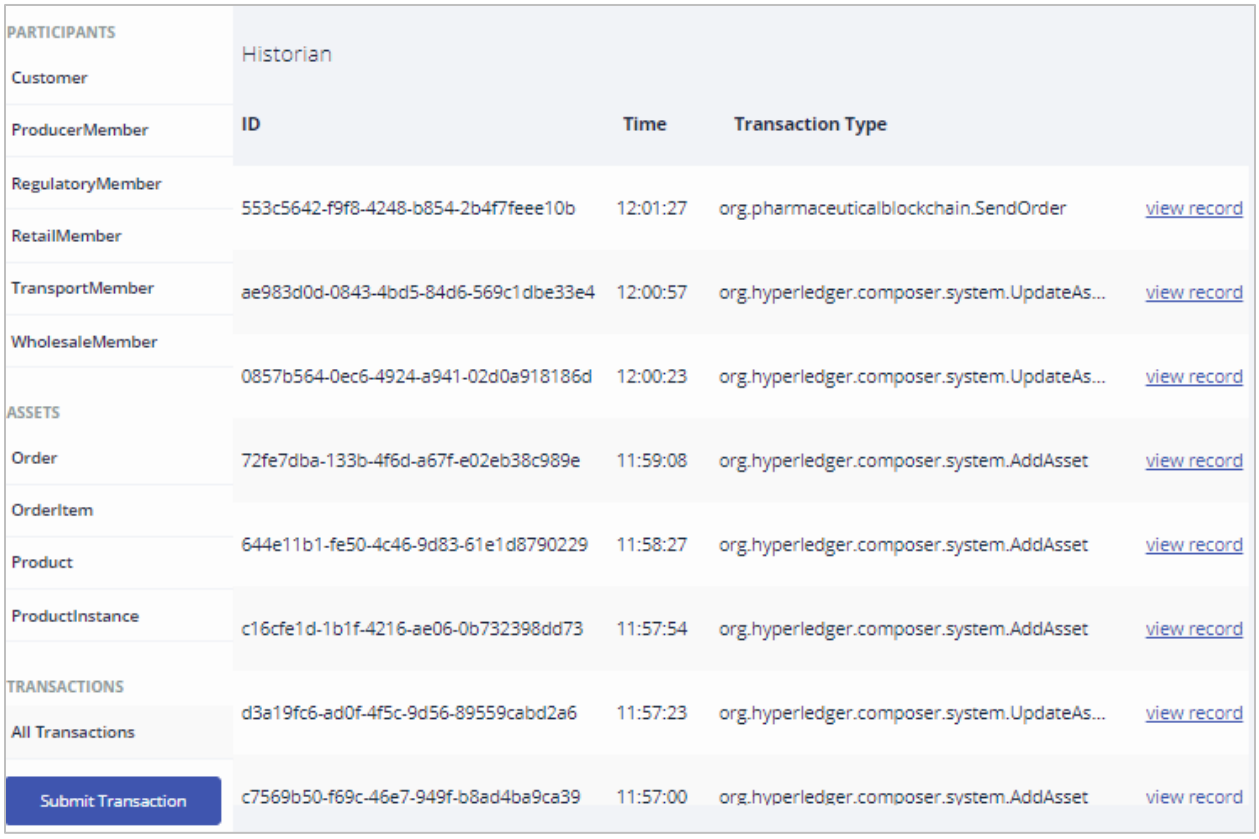
ShipOrder operacija predstavlja operaciju koja se pokreće prilikom završetka pakovanja robe u skladu sa narudžbenicom. Operacija podrazumeva promenu statusa instanci proizvoda (u fazi dostave) koji se šalju. Pored toga, poslate instance proizvoda smeštaju se u kolekciju dolazećih instanci (*incomingProductInstances*) proizvoda naručioca.

DeliverOrder bi izvršavala posebna kurirska služba ili zaposleni u kompaniji od koje su i naručeni proizvodi. U ovom koraku se stanje računa naručioca umanjuje za iznos narudžbenice, a za isti taj iznos se uvećava stanje na računu pošiljaoca. Pored toga, menja se vlasnik instanci proizvoda. Nakon ove operacije stanje narudžbenice se menja tako da označava da je narudžbenica obrađena u potpunosti.

Prava pristupa su definisana u posebnom fajlu. Svaki akter mreže ima uvid u klase proizvoda, a veleprodajne kompanije mogu davati posebne cene za određene partnere. Proizvođači ili uvoznici mogu da stave novi proizvod u promet, a regulatorna tela mogu da ga menjaju, odnosno da daju ili povuku dozvolu za proizvod. Svaka kompanija može da kreira narudžbenicu. Uvid u konkretnu narudžbenicu imaju samo pošiljalac i primalac.

Kao što je pomenuto, razvoj i testiranje poslovne mreže znatno olakšava playground composer. Kroz njegov grafički interfejs je omogućeno jednostavno pozivanje transakcija, uključujući i dodavanje, izmenu i brisanje digitalnih dobara i učesnika u sistemu.

Composer Playground poseduje aspekt koji se tiče definicije poslovne mreže gde se specificiraju modeli, a postoji i aspekt za testiranje poslovne mreže. Početni prikaz u playground-u kada je reč o testiraju obuhvata 2 celine (Slika 17). Na levom delu prozora se nalazi lista učesnika i digitalnih dobara. Klik na stavku iz liste će promeniti desni deo prozora tako da prikaže listu odgovarajućih zapisa. Pored toga moguće je videti i sve transakcije izvršene u sistemu klikom na *All Transactions* dugme pri dnu strane. Ovaj prikaz omogućuje uvid u id transakcije (jedinstveni identifikator), vrstu transakcije, vreme pozivanja transakcije kao i detalje o promenama podataka izazvanim pozivom transakcije (Slika 17).

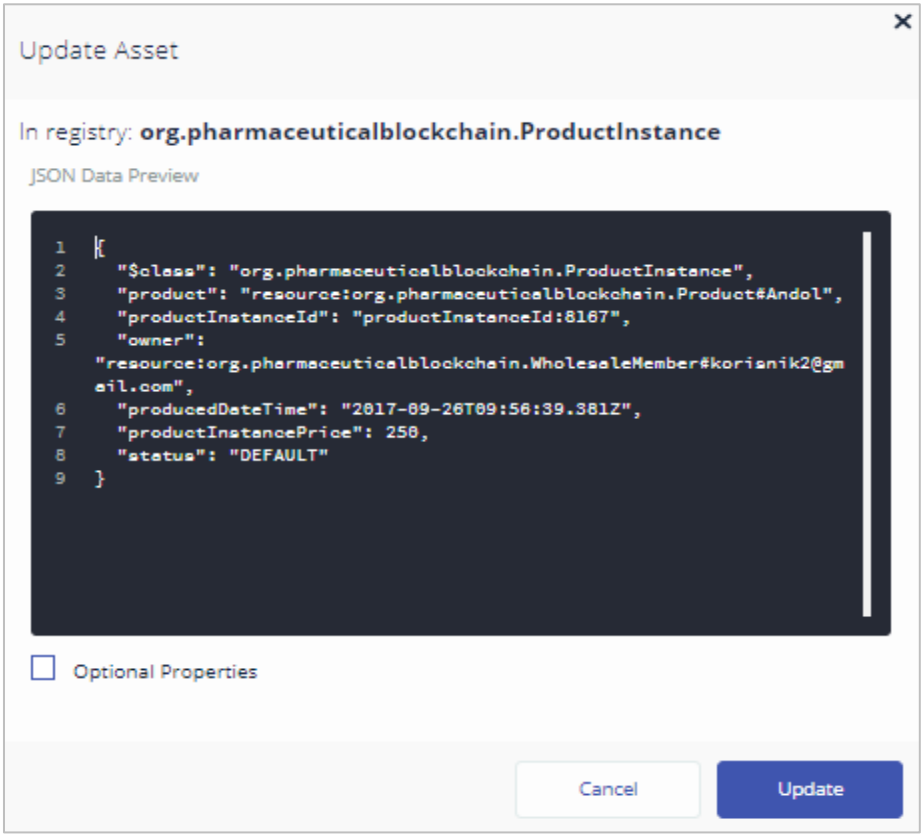


Slika 17. Listing transakcija u Playground-u

Prikaz stavki narudžbenice je prikazan na slici 18. Pored svake stavke se nalazi dugme za njenu izmenu i brisanje. U desnom gornjem uglu (Slika 18) nalazi se dugme (Create New Asset) na čiji se klik otvara modalna forma za dodavanje novog učesnika u sistem.

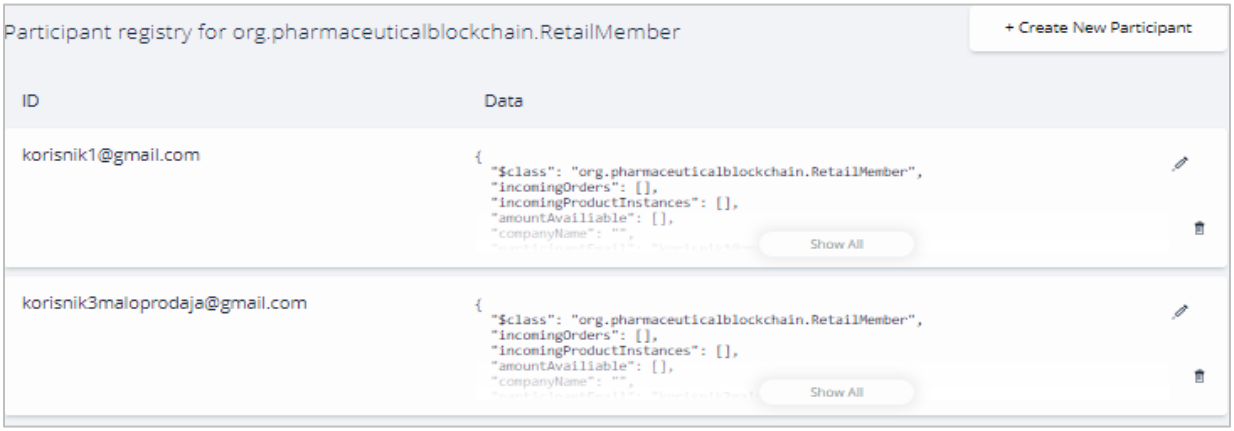
Slika 18. Lista stavki narudžbenice u Composer Playground-u

Composer playground koristi modalne prozore za unos i izmenu podataka. Modalna forma za unos ili izmenu instance proizvoda u JSON obliku je prikazana na slici 19.



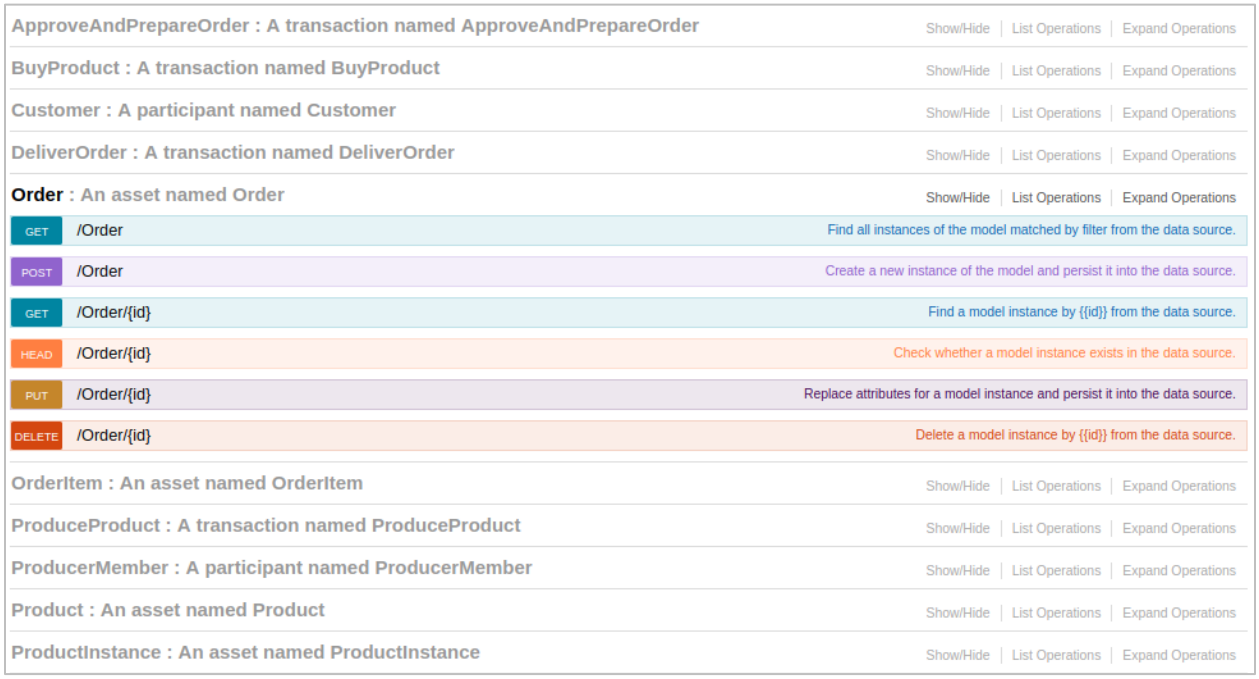
Slika 19. Dodavanje nove instance proizvoda kroz Composer Playground

Grafički interfejs za prikaz, dodavanje, izmenu i brisanje učesnika je sličan kao i u slučaju digitalnih dobara. Prikaz svih korisnika iz grupe maloprodajnih korisnika je dat na slici 20.



Slika 20. Listing korisnika maloprodaje

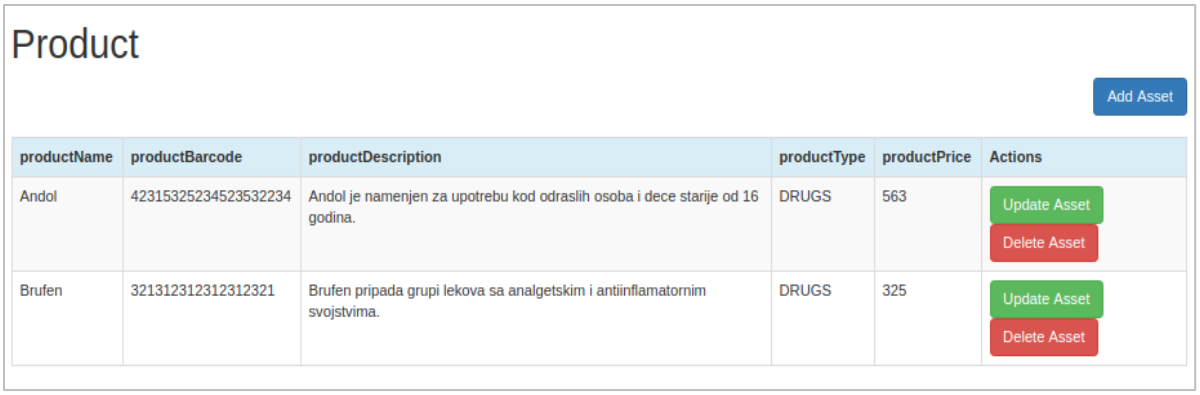
Generisana arhiva poslovne mreže (.bna fajl) takođe može biti iskorišćena za generisanje REST API-a, i na taj način omogućiti brz razvoj serverskog dela *web* aplikacija. Pomenute *web* aplikacije služe kao klijentske aplikacije za komunikaciju sa Hyperledger Fabric mrežom.



Slika 21. Rest putanje za rad sa poslovnom mrežom

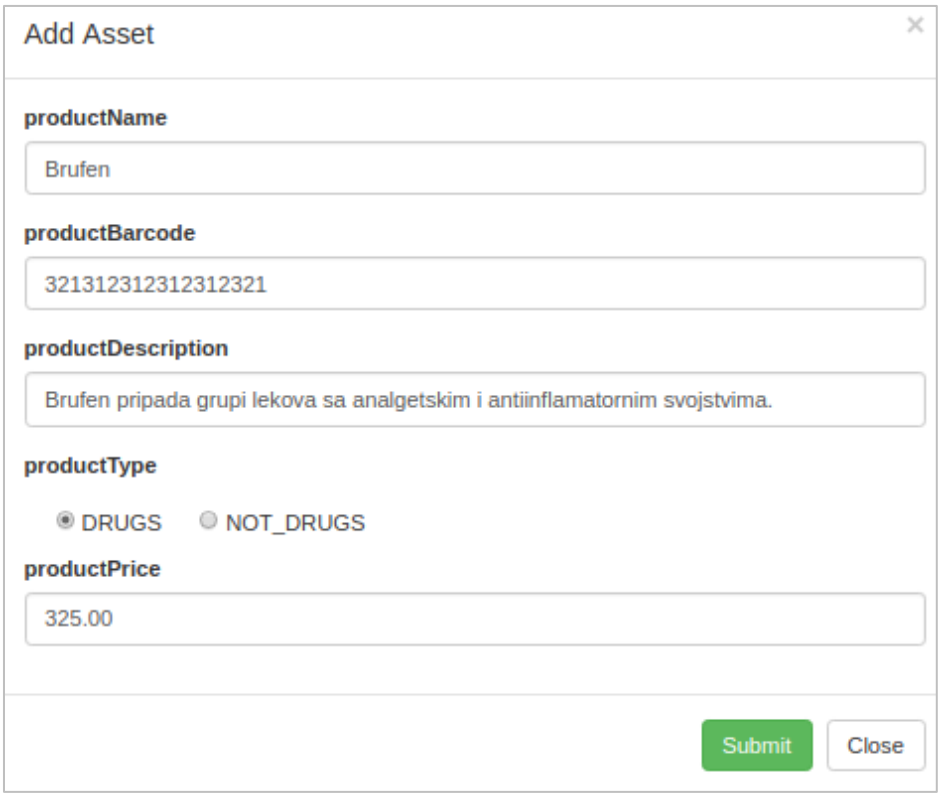
Za svaki tip učesnika odnosno digitalno dobro se kreira skup rest putanja (Slika 21). Na slici 21. su takođe prikazane REST metode koje omogućavaju rad sa narudžbenicom (*Order*). Na osnovu definicije poslovne mreže generisan je i kostur *web* aplikacije korišćenjem Yaoman alata. Generisan je kod za AngularJS2 radni okvir, a komunicira sa serverskim delom preko generisanog REST API-ja. Aplikacija koja je generisana na pomenuti način trenutno ne podržava dodavanje novih učesnika u mrežu kroz grafički interfejs. Aplikacija omogućuje sve CRUD opreracije nad digitalnim dobrima, i poseduje jednostavan grafički interfejs za pozivanje istih.

Na slici 22. je dat prikaz klasa proizvoda na grafičkom intefejsu aplikacije. Ovaj prikaz omogućuje i brisanje odnosno izmenu klase proizvoda.



Slika 22. Lista proizvoda u generisanoj aplikaciji

Dodavanje nove klase proizvoda se vrši klikom na Add Asset čime će se otvoriti modalna forma za unos podataka o novoj klasi proizvodu (Slika 23).



Slika 23. Forma za dodavanje klase proizvoda

Forma sadrži mogućnost unosa svih relevantnih podataka o entitetu. Tipovi nabrajanja su na grafičkom interfejsu prikazani kao polja izbora. Primarni ključ entiteta nije prikazan na formi

jer on predstavlja negovoreću šifru (niz karaktera) koja se automatski generiše prilikom unosa entiteta u sistem.

5. Zaključak

Blockchain tehnologija je relativno nova i još uvek ne postoji dovoljno razvijena baza znanja, iskustva i relevantan broj uspešnih implementacija. Radovi iz ove oblasti su često samo konceptualni, bez vizije konkretne primene. Oni sadrže veliki broj informacija, ali su one često uopštene, nepotpune ili neproverljive. Trenutno tipična primena ove tehnologije jeste Bitcoin mreža koja se opisuje kao: decentrazlizovana *peer-to-peer* mreža koja služi za slanje novčanih transakcija, gde je klasičan novac zamenjen digitalnim reprezentom (Bitcoin). Postojeći model razmene je preskup i nepouzdan i kao takav je godinama unazad u krizi i ne odgovara savremenom načinu placanja. U jednom trenutku će verovatno doći do uvođena svetske digitalne monete ili više njih. Digitalne valute su samo jedna, trenutno najatraktivnija implementacija blokchaina, ali se ovaj koncept potencijalno može primeniti na širok skup oblasti u industriji. Zato je veliki broj aktera zaintesovan za razvoj ove oblasti: velike kompanije, poput IBM-a koje se bave konceptima i razvojem okruženja, manje kompanije koje se spremaju za implementaciju, vlade širom sveta kao i vojne strukture. Pored toga veliki broj ljudi pokušava da zaradi "proizvodeći" digitalne valute, dok proizvođači hardvera prodaju enormne količine grafičkih kartica koje služe toj svrsi. Takođe, sve više ljudi se pridružuje razvoju ove tehnologije i aktivno razmenjuju iskustva u zajednici otvorenog koda, uz moderaciju velikih kompanija i naučne zajednice. Ovo jeste oblast koja može doneti promene, ali takođe i velike potrese, s obzirom da se najveći broj novčanih transakcija obavlja u paritetu sa američkim dolarom, koji nema zlatnu podlogu, a produkuje ga Američka centralna banka (Fed) čiji rad se odavno ne smatra kredibilnim.

U ovom radu opisane su osnove blockchain distribuiranih sistema i predlog njihove primene za podršku lancu nabavke u farmaceutskoj industriji. Proces praćenja farmaceutskih proizvoda je složen, jer postoji veliki broj učesnika u ovom procesu. Praćenje je takođe otežano činjenicom da se lekovi uvoze, kao i da se nalaze na različitim listama prodaje, sa ili bez subvencije. Lekovi se proizvode u serijama, sa različitim rokovima upotrebe što je takođe značajno za proces praćenja. Deo lekova se nalazi u slobodnoj prodaji, dok se deo može kupiti samo uz recept zdravstvene ustanove.

Saradnja maloprodajnih i veleprodajnih kompanija i proizvođača je vrlo intenzivna. Ona uključuje razmenu velikog broja dokumenata i sklona je greškama zbog zastarelog načina komunikacije. Pomenute poteškoće u lancu nabavke farmaceutskih proizvoda bi mogle biti rešene korišćenjem blockchain tehnologija. Stoga, prezentovano idejno rešenje je zasnovano na Fabric blockchain-u i moglo bi da pomogne u realizaciji adekvatnog modela celovite razmene i obrade podataka.

Blockchain tehnologija je mlada i sigurno je da će njen razvoj i njena primenljivost napredovati u narednim godinama. Blockchain tehnologija takođe može uticati na dalji razvoj kriptografije, neuronskih mreža i veštacke inteligencije jer sadrže slične elemente. Prilikom istraživanja opisanog u ovom radu, pokazalo se da je Hyperledger Fabric platforma pogodna za primenu u farmaceutskoj oblasti. Nije jednostavno pokrenuti i testirati Fabric mrežu, što je

otežavalo razvoj praktičnog rešenja. Razvoj kroz Composer omogućuje programerima da se fokusiraju na implementaciju poslovne logike, ali takav razvoj ipak ima određena ograničenja. Ona se uglavnom ogledaju u manjoj kontroli nad mrežom i nemogućnosti specifikacije svih aspekata mreže, kao što su pravila prihvatanja transakcija. Fabric Composer jeste koristan alat koji omogućuje brz razvoj konceptualnih rešenja za Fabric platformu. Ipak u ovom trenutku, za razvoj produkcionih rešenja je potrebno koristiti direktno Fabric platformu i implementirati pametne ugovore u jeziku Go.

Kao budući rad, planirano je proširivanje funkcionalnosti razvijene aplikacije i njen kompletan razvoj direktno nad Fabric platformom, bez posredstva Fabric Composer-a. Dalji razvoj aplikacije bi uključivao mogućnosti dodavanja učesnika u sistem kroz grafički interfejs i poboljšavanje istog. Validacija podataka u ovom trenutku postoji samo na serverskom (engl. *backend*) delu aplikacije, tako da je u u budućem radu potrebno implementirati validaciju i na prednjem delu (engl. *frontend*) aplikacije. Pored toga, potrebno je napraviti posebne forme za kreiranje i pozivanje transakcija nad Fabric blockchain mrežom. Što se modela poslovne mreže tiče, planirano je da se proširi skup učesnika kako bi se podržale bolnice, a da se postojećim učesnicima pruže dodatne mogućnosti.

Literatura

[1] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, October, 2008.

[2] Proof of work, dostupno na: https://en.bitcoin.it/wiki/Proof_of_work, poslednji pristup 30.8.2017.

[3] Proof of stake, dostupno na: https://en.wikipedia.org/wiki/Proof-of-stake poslednji pristup 30.8.2017.

[4] Fabric Composer Documentation, dostupno na: https://hyperledger.github.io/composer/, poslednji pristup 20.9.2017.

[5] R. C. Merkle, Protocols for public key cryptosystems, in *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pp. 122-133, April, 1980.

[6] M. Castro and B. Liskov, Practical Byzantine fault tolerance, in *Proc. 3rd OSDI*, pp. 173–186, February, 1999.

[7] Hyperledger Fabric Documentation, dostupno na: https://hyperledger-fabric.readthedocs.io/en/latest/, poslednji pristup 24.9.2017.

[8] L. Lamport, R. Shostak, M. Pease, The Byzantine Generals Problem, *ACM Transactions on Programming Languages and Systems*. 4(3): pp. 382-401, July, 1982.

[9] M. Swan, *Blockchain: Blueprint for a New Economy*, O'Reilly Media, Sebastopol, CA, USA, 2015.

[10] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Crypto-Currencies*, O'Reilly Media, Sebastopol, CA, USA, 2014.

[11] S. Haber, W. S. Stornetta, How to time-stamp a digital document, *Journal of Cryptology*, vol 3, no 2, pp 99-111, 1991.

[12] The difference between a Private, Public & Consortium Blockchain, dostupno na: http://www.blockchaindailynews.com/The-difference-between-a-Private-Public-Consortium-Blockchain_a24681.html, poslednji pristup 21.9.2017.

[13] Hyperledger Project, dostupno na: https://www.hyperledger.org/, poslednji pristup 24.9.2017.

[14] A. Back, Hashcash - a denial of service counter-measure, dostupno na: http://www.hashcash.org/papers/hashcash.pdf, poslednji pristup 15.7.2017

[15] World Health Organization - Growing threat from counterfeit medicines, dostupno na: http://www.who.int/bulletin/volumes/88/4/10-020410/en/, poslednji pristup 22.9.2017.

[16] PharmExec.com - Fake Medications, Real Solutions, dostupno na: http://www.pharmexec.com/fake-medications-real-solutions,poslednji pristup 22.9.2017.

[17] HashCash, dostupno na: https://en.bitcoin.it/wiki/Hashcash, poslednji pristup 28.8.2017.

[18] PeerCoin, dostupno na: https://en.wikipedia.org/wiki/Peercoin, poslednji pristup 29.8.2017.

[19] Nick Szabo, Smart Contracts: Building Blocks for Digital Markets, dostupno na: www.fon.hum.uva.nl, poslednji pristup 5.8.2017.

[20] A. Narayanan, J. Bonneau, E. Felten, A. Miller, S. Goldfeder, *Bitcoin and Cryptocurrency Technologies*, Princeton University, Princeton, New Jersey, USA , 2016

[21] Coinbase, dostupno na: https://www.coinbase.com/, poslednji pristup: 13.9.2017.

[22] SHA-2, dostupno na: https://en.wikipedia.org/wiki/SHA-2, poslednji pristup 25.8.2017.

[23] Avalanche effect, dostupno na: https://en.wikipedia.org/wiki/Avalanche_effect, poslednji pristup 25.8.2017.

[24] Bitcoin Dificulty, dostupno na: https://en.bitcoin.it/wiki/Difficulty, poslednji pristup 28.8.2017.
[25] Mayority attack, dostupno na: https://en.bitcoin.it/wiki/Majority_attack, poslednji pristup 15.9.2017.
[26] Satoshi Nakamoto, dostupno na : https://en.wikipedia.org/wiki/Satoshi_Nakamoto, poslednji pristup 30.8.2017.
[27] History of Bitcoin, dostupno na: https://en.wikipedia.org/wiki/History_of_bitcoin, poslednji pristup 30.8.2017.
[28] How big is Bitcoin, dostupno na: https://news.Bitcoin.com/how-big-is-Bitcoin/, poslednji pristup 5.9.2017.
[29] Why the Blockchain and the Bitcoin Wallet Balances Differ, dostupno na: https://blog.barthe.ph/2014/04/03/bitcoin-balance-differs/, poslednji pristup 17.9.2017.
[30] IBM.com - Top 6 technical advantages of Hyperledger Fabric for blockchain networks, dostupno na: https://www.ibm.com/developerworks/cloud/library/cl-top-technical-advantages-of-hyperledger-fabric-for-blockchain-networks/index.html, poslednji pristup 20.9.2017.
[31] Wikipedia – Database, dostupno na: https://en.wikipedia.org/wiki/Database, poslednji pristup 20.9.2017.
[32] Blockchain technology – beyond Bitcoin, dostupno na: http://scet.berkeley.edu/wp-content/uploads/BlockchainPaper.pdf, poslednji pristup 20.9.2017.
[33] openwt.com - Centralization vs decentralization, dostupno na: https://openwt.com/sites/default/files/centralization_vs_decentralization-smaller.png, poslednji pristup 20.9.2017.
[34] How blockchain works, dostupno na: https://blogs.thomsonreuters.com/answerson/wp-content/uploads/sites/3/2016/01/infographic-how-blockchain-works.jpg, poslednji pristup 20.9.2017.
[35] openwt.com - Blockchain, dostupno na: https://openwt.com/en/trends/blockchain, poslednji pristup 20.9.2017.
[36] bitcoin-class.org , dostupno na: http://bitcoin-class.org/classes/class7/merkle.png, poslednji pristup 20.9.2017.
[37] draglet.com – Smart Contract Application Examples and Use Cases, dostupno na: https://www.draglet.com/blockchain-applications/smart-contracts/use-cases, poslednji pristup 20.9.2017.
[38] bitcoin.com – Bitcoin Developer Guide, dostupno na: https://bitcoin.org/en/developer-guide, poslednji pristup 20.9.2017.

Prilog 1: Spisak korišćenih termina

Pojam	Značenje
CRUD operacije	Operacije kreiranja, čitanja, menjanja i brisanja podataka
REST	REST (REpresentational State Transfer) je arhitekturalni stil koji se koristi za razvoj veb aplikacija, razvijan je paralelno sa HTTP 1.1.
HEŠ	Vrednost koju heš transformacija daje na izlazu.
Peer-to-peer mreža	Mreža ravnopravnih članova
AngularJS2	JavaScript Framework koja omogućava razvoj i testiranje klijentskog dela veb aplikacije.
Radni okvir (Framework)	Predstavlja softverski alat koji predviđa određen način rada. Koristi se da pojednostavi proces razvoja softvera.
Modalna forma	Prozor sa poljima za unos koji blokira rad aplikacije dok god se ne zatvori.

Prilog 2: Spisak tabela

Tabela 1. Razlike između privatnih i javnih blockchain mreža 5
Tabela 2. Razlike Hyperledger-a i Bitcoin-a 24

Prilog 3: Spisak slika

Slika 1. Propagacija transakcije kroz blockchain 3
Slika 2. Distribuirani i centralizovani pristup 4
Slika 3. Tradicionalni i blockchain pristup 6
Slika 4. Merkle stablo 7
Slika 5. Lanac blokova 8
Slika 6. Male promene na ulazu u SHA256 rezultuju značajno različitim izlazom 10
Slika 7. Princip rada pametnog ugovora 12
Slika 8. Primer *Query* i pomoćne *read* metode 17
Slika 9. Primer *Invoke* i pomoćne metode *write* 18
Slika 10. Tok transakcije u Hyperledger Fabric-u 20
Slika 11. Ulazi i izlaz u Bitcoin transakciji 22
Slika 12. Promena vrednosti Bitcoin-a sa vremenom 23
Slika 13. Arhiva poslovne mreže u Fabric Composer-u 28
Slika 14. Hijerarhija učesnika u sistemu 30
Slika 15. Model učesnika u sistemu 31
Slika 16. Model digitalnih dobara u Composer-u 32
Slika 17. Listing transakcija u Playground-u 35
Slika 18. Lista stavki narudžbenice u Composer Playground-u 35
Slika 19. Dodavanje nove instance proizvoda kroz Composer Playground 36
Slika 20. Listing korisnika maloprodaje 36
Slika 21. Rest putanje za rad sa poslovnom mrežom 37
Slika 22. Lista proizvoda u generisanoj aplikaciji 38
Slika 23. Forma za dodavanje klase proizvoda 38

Biografija

Aleksa Mirković je rođen 3. novembra 1993. godine u Šapcu, Srbija. Završio je Gimnaziju u Šapcu, informatički smer. Školske 2012/2013. godine upisao je Fakultet tehničkih nauka u Novom Sadu, odsek Računarstvo i automatika, usmerenje Primenjene računarske nauke i informatika. Osnovne akademske studije završio je školske 2016/2017. godine i iste godine je upisao master studije na Fakultetu tehničkih nauka. Položio je sve ispite propisane planom i programom.