УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Марина Ненић

Blockchain дистрибуирани системи и њихова примјена у индустрији видео игара

Мастер рад

Нови Сад, 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**:		Марина Ненић
Ментор, **МН**:		др Душан Гајић, доцент
Наслов рада, **НР**:		*Blockchain* дистрибуирани системи и њихова примјена у индустрији видео игара
Језик публикације, **ЈП**:		Српски / ћирилица
Језик извода, **ЈИ**:		Српски/енглески
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)		5/64/0/2/36/0/1
Научна област, **НО**:		Електротехника и рачунарство
Научна дисциплина, **НД**:		Примењене рачунарске науке и информатика
Предметна одредница / кључне речи, **ПО**:		Базе података, дистрибуирани информациони системи, *blockchain*
УДК		
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		У овом раду описани су основи концепти *blockchain* технологије и једне од њених познатијих имплементација, *Hyperledger Fabric*-а. Такође, описана је и апликација имплементирана на *Hyperledger Fabric* платформи, развијена са циљем управљања ресурсима унутар видео игара.
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**:		Marina Nenić	
Mentor, **MN**:		Dušan Gajić, Ph. D	
Title, **TI**:		Blockchain Distributed Systems and Their Application in the Video Games Industry	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian/English	
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)		5/64/0/2/36/0/1	
Scientific field, **SF**:		Electrical and computer engineering	
Scientific discipline, **SD**:		Applied computer science and informatics	
Subject/Key words, **S/KW**:		Databases, distributed information systems, blockchain	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad	
Note, **N**:			
Abstract, **AB**:		This thesis presents the fundamentals of the blockchain technology and one of its popular implementations, Hyperledger Fabric. It also describes an application implemented using the Hyperledger Fabric platform, which is developed for asset management in video games.	
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: 4/64

(Податке уноси предметни наставник - ментор)

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

Студент:	Марина Ненић	Број индекса:	Е2 11/2016
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 ПРИМЈЕНА У ИНДУСТРИЈИ ВИДЕО ИГАРА**

ТЕКСТ ЗАДАТКА:

- Проучити основне концепте и механизме *blockchain* дистрибуираних система.
- Упознати се са *Hyperledger Fabric* платформом, као примјером конкретне имплементације *blockchain*-а.
- Упоредити *Hyperledger Fabric* и *Ethereum* имплементације *blockchain* система.
- Упознати се са *Hyperledger Composer* алатом за развој пословних апликација базираних на *Hyperledger Fabric* платформи.
- Специфицирати модел *blockchain* мреже намјењене за управљање ресурсима унутар видео игара, који садржи дефиниције ресурса, учесника и операција које учесници могу извршавати над ресурсима.
- На основу специфицираног модела имплементирати одговарајуће софтверско рјешење засновано на *Hyperledger Fabric*-у.

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

Примерак за: ☐ - Студента; ☐ - Ментора

Садржај

1. УВОД ... 1
2. ОСНОВЕ *BLOCKCHAIN* СИСТЕМА ... 3
   2.1. *Peer-to-peer* архитектура ... 3
      2.1.1. Синхронизација дневника трансакција ... 4
      2.1.2. Типови чворова ... 5
      2.1.3. Типови мрежа ... 6
   2.2. Структура отпорна на измјене ... 7
   2.3. Консензус ... 11
      2.3.1. Практична византијска толеранција грешке (енгл. *Practical Byzantine Fault Tolerance*) ... 12
      2.3.2. Доказ посла (енгл. *Proof of Work*) ... 12
      2.3.3. Доказ посједовања (енгл. *Proof of Stake*) ... 14
   2.4. Паметни уговори ... 17
3. *HYPERLEDGER FABRIC* ... 19
   3.1. Пројекат *Hyperledger* ... 19
   3.2. *Hyperledger Fabric* ... 21
      3.2.1. Сервиси чланства ... 22
      3.2.2. *Chaincode* ... 23
      3.2.3. Организација података ... 24
      3.2.4. Консензус ... 24
   3.3. Поређење *Hyperledger Fabric*-а и *Ethereum*-а ... 27
      3.3.1. Циљни корисници ... 27
      3.3.2. Консензус и повјерљивост ... 27
      3.3.3. Криптовалута ... 28
4. ОПИС СОФТВЕРСКОГ РЈЕШЕЊА ... 29
   4.1. Опис коришћених алата и технологија ... 29
   4.2. Опис *blockchain* мреже ... 31
5. ЗАКЉУЧАК ... 44

ЛИТЕРАТУРА ... 46

ДОДАТАК А ... 48

СПИСАК КОРИШЋЕНИХ СКРАЋЕНИЦА ... 54

СПИСАК СЛИКА ... 55

СПИСАК ТАБЕЛА ... 57

СПИСАК ЛИСТИНГА ... 58

БИОГРАФИЈА ... 59

1. УВОД

Основни механизам размјене добара као и самог пословања чине трансакције. Трансакција у пословном смислу представља договор између двије странке, најчешће купца и продавца, о размјени материјалних добара, услуга, као и финансијских средстава. Читаво савремено пословање заснива се на том концепту. Информативна вриједност пословних трансакција је изузетно велика јер трансакције посједују информације потребне за утврђивање власништва над добрима, одговорности појединаца, увид у стање финансијских средстава, а такође пружају и увид у историју активности пословања те је њихово записивање и чување од кључног значаја.

Са напретком технологије све већи дио људских активности почиње да се заснива на употреби рачунара који постају незамјењиви у пословању и свакодневном животу па тако и трансакције добијају свој електронски облик. Ту на сцену ступају базе података као централно мјесто за складиштење трансакција и других података битних за пословање.

Дуги низ година у информационим системима доминирале су централизоване базе података, односно базе код којих се сви подаци чувају на једном централном мјесту. Предност организације података на једном мјесту огледа се у томе што су такве базе лакше за управљање и одржавање, али то са собом носи и одређене недостатке. Један од главних недостатака представља опасност од пада система. Немогућност приступа подацима у централизованој бази значио би престанак функционисања читавог информационог система. Рјешење овог проблема пронађено је у децентрализацији. Међутим јавља се још један битан проблем који децентрализација не може да ријеши а то је посредништво. У већини случајева пословне трансакције треба да прођу низ корака да би се утврдила тачност информација и да би биле потврђене као валидне. Потребно је пронаћи организацију од повјерења која би била задужена за такве провјере, што значи да се јавља потреба за посредником. Одличан примјер посредника јесу банке. Посредништво уводи додатне кораке у процесе пословања који захтијевају додатан утрошак времена и средстава.

Због поменутих проблема који значајно отежавају процес пословања јавља се потреба за новим начином организације и чувања података и трансакција. Јавља се потреба за технологијом која би обезбиједила већу доступност података и укинула посредништво али истовремено задржала неопходне функционалне карактеристике као што су сигурност, брзина и отпорност на злоупотребе. Технологија *blockchain*-а (енгл. *blockchain – ланац блокова*) управо испуњава све претходно наведене захтјеве. Иако је још увијек нова, ова технологија уноси значајне промјене у досадашњи приступ реализацији трансакција, чини их приступачијим и јавно провјерљивим. Дистрибуирана природа омогућава лако креирање ефикасних и економичих пословних мрежа које не захтијевају централно контролно мјесто (банке, државе) као што је случај са досадашњим начином пословања.

Циљ овог рада јесте упознавање са *blockchain* технологијом и њеним најпознатијим имплементацијама, као и примјена исте у индустрији видео игара, са акцентом на *Hyperledger Fabric* имплементацији *blockchain*-а [1]. За испуњење поменутог циља потребно је реализовати неколико корака. Најприје је потребно извршити анализу теоријских основа *blockchain* технологије. Ту се прије свега мисли на концепте и механизме који омогућавају *blockchain* технологији да испуни претходно наведене захтјеве приступачности, сигурности и отпорности. Следећи корак је избор једне од имплементација *blockchain*-а која ће бити основ за изградњу софтверског рјешења. У овом раду изабран је *Hyperledger Fabric*. На крају је потребно имплементирати софтверско рјешење које ће служити за управљање ресурсима, односно куповину и размјену добара, у видео играма.

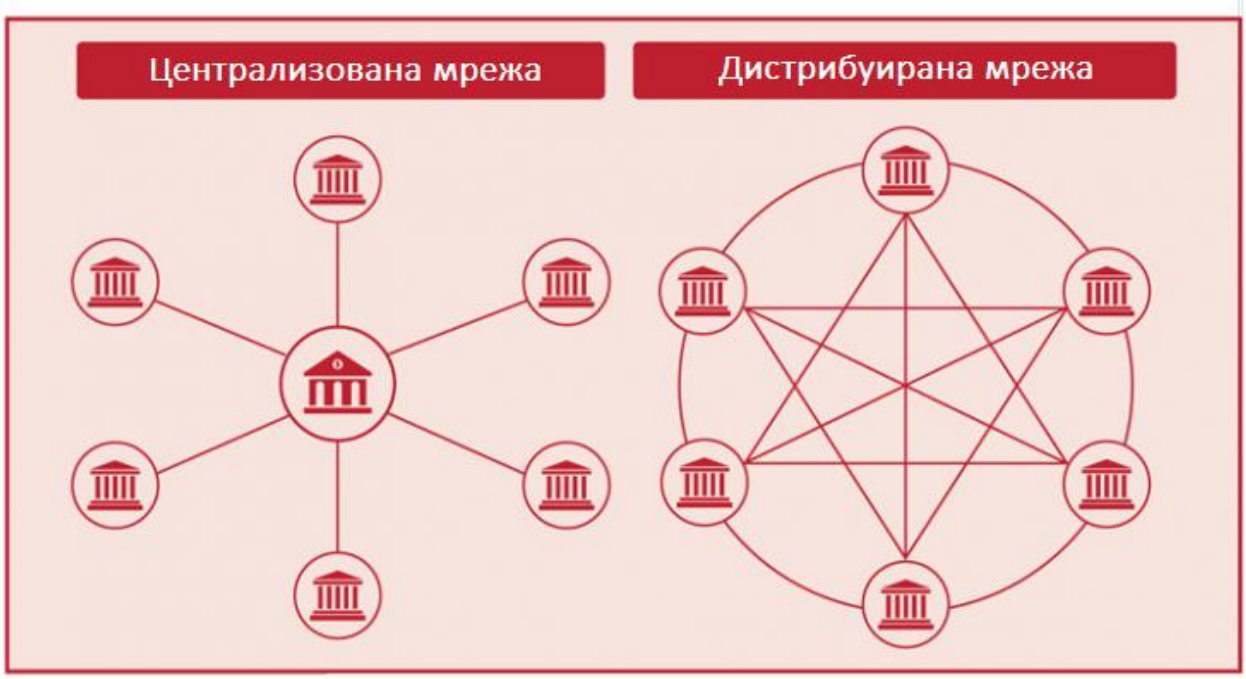
Након уводног поглавља слиједи поглавље „Основе *blockchain* система“ у ком су објешњени основи појмови *blockchain* технологије. Након тога слиједи поглавље „*Hyperledger Fabric*“ у којем су представљена једна од значајнијих имплементација *blockchain*-а, *Hyperledger Fabric*, чији је циљ проширење спектра примјене *blockchain* технологије у индустрији. Затим слиједи опис софтверске имплементације система који је дат у поглављу „Опис софтверског рјешења“ гдје су детаљно приказане имплементације паметних уговора и корисничке апликације. Након тога, у поглављу „Закључак“, дат је осврт на резултате рада и уочене недостатке, као и планови за даљи рад.

2. ОСНОВЕ *BLOCKCHAIN* СИСТЕМА

Blockchain је врста реплициране дистрибуиране базе података која чува дигиталне трансакције [2,3]. Организована је у форми једноструко спрегнуте листе (ланац), гдје улогу чворова имају блокови који садрже податке о трансакцијама заштићеним криптографским методама. Из такве структуре потиче назив *blockchain* [4]. База је реплицирана на мрежи који је заснована на *peer-to-peer* архитектури.

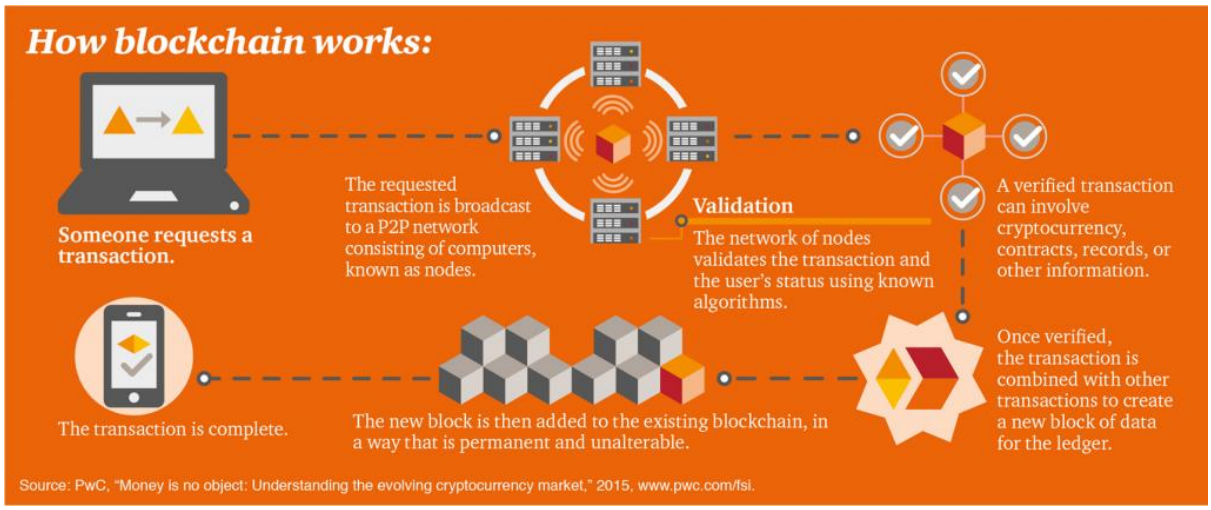
2.1. *Peer-to-peer* архитектура

Peer-to-peer архитектура представља дистрибуирану архитектуру у којој су сви чланови равноправни, комуницирају путем интернета и уједно представљају и снабдијеваче и потрошаче ресурса [5]. Чињеница да су сви чланови равноправни спрјечава једног или одређен број различитих учесника, све док је тај број довољно мали, да угрозе или контолишу мрежу. Преузимање већинског броја чворова у мрежи је изузетно компликован и скуп подухват те би напори за то далеко превазишли напоре потребне да се угрози централизован систем па се из тог разлога *blockchain* технологија сматра безбједнијом [2]. На слици 1 приказане су илустрације дистрибуиране и централизоване мреже [6].



Слика 1. Централизована и дистрибуирана мрежа

Чворови (чланови мреже) који учествују у реализацији трансакција посједују копије свих записа, те стално међусобно комуницирају и синхронизују записе. Такав начин функционисања омогућава реализацију дигиталних трансакција без посредника. Да би се трансакција извршила и била уписана у *blockchain* дневник трансакција (енгл. *ledger)* потребно је послати односно објавити трансакцију осталим члановима мреже. Члан који је иницијатор трансакције шаље трансакцију свим члановима на мрежи са којима је повезан. Сваки од чланова приликом пријема трансакције извршава низ провјера да би утврдио да је иста валидна. Све валидне трансакције које прими одређени члан смјештају се у његов скуп трансакција које чекају да буду додате у *blockchain* приликом креирања наредног блока. У колико трансакција успјешно прође све провјере, дати члан провјерава да ли је та иста трансакција већ претходно стигла до њега и на основу тога доноси одлуку да ли да је прослиједи даље члановима са којима је он повезан. Завхаљујући томе сваки члан само једном просљеђује даље одређену трансакцију. На слици 2 [7] приказан је ток трансакције од иницијализације до уписа у *blockchain* ланац.



Слика 2. Ток транцакције у blockchain мрежи

2.1.1. Синхронизација дневника трансакција

Када је у питању синхорнизација дневника трансакција различите имплементације користе различите приступе. У једној од најпознатијих имплементација, *Bitcoin*-у [2], члан мреже који први ријеши одређени компјутерски проблем добија право додавања новог блока у ланац. Сви остали чланови када виде новододати блок провјеравају свој скуп трансакција које чекају да буду додате у *blockchain*. Пошто у мрежи између чворова постоји одређено кашњење може се десити да различити чланови имају различите скупове трансакција на чекању. У колико се деси да је нека трансакција из датог скупа у сукобу са трансакијама из новообјевљеног блока, дати члан ће избацити ту трансакцију из свог скупа јер више не важи као валидна. Уколико се у скупу трансакција налази иста трансакција као и у новом блоку, дати члан ће примјенити ту трансакцију на своју копију базе уколико је потребно (неке имплементације *blockchain*-а поред дневника трансакција чувају и тренутно стање

ресурса у посебној бази) или је само избацити из скупа јер дата трансакција више не чека да буде објављена него се већ налази у ланцу. Све остале трансакције који нису ни у сукобу а ни објављене остају у скупу и чекају следећи блок.

Иако peer-to-peer архитектура *blockchain*-а налаже да сви чворови у мрежи буду равноправни, могу постојати чворови са различитим улогама. Улоге чворова зависе од функционалних захтијева које се јављају у датој области примјене.

2.1.2. Типови чворова

Ако посматрамо *blockchain* са становишта електронских валута, ту се издваја неколико различитих врста чворова. Пуни чворови (енгл. *full nodes*) су чворови који одржавају комплетну копију дневника трансакција у својој меморији и могу да провјеравају трансакције без икакве спољне помоћи. Да би могао да обавља улогу пуног чвора, чвор мора бити константно присутан на мрежи и ослушкивати трансакције. Задатак пуних чворова јесте да одржавају мрежу и воде рачуна да у ланац доспију само валидне трансакције. Са друге стране постоје лагани или танки чворови (енгл. *lightweight nodes*) који чувају и прате само оне податке и блокове који се односе на њима битне трансакције. Танки чворови се ослањају на пуне када је у питању сигурност и валидност осталих трансакција у ланцу [3]. Поред пуних и танких чворова постоје и чворови рудари (енгл. *mineres*) који су задужени за додавање нових блокова у ланац. Такви чворови посједују доста рачунарских ресурса јер додавању новог блока претходи рачунарски интензиван задатак који треба да ријеше. Такав приступ назива се доказ посла и његов главни циљ јесте да обезбиједи демократичност и насумичност у процесу додавања блокова, односно да пружи могућност сваком члану рудару да баш његов блок буде следећи у *blockchain* ланцу.

Поред електорнских валута *blockchain* технологија је нашла примјену и у области индустрије. Најпознатија имплементација *blockchain*-а која се усмјерила на задовољавање широких захтијева индустрије јесте *Hyperlegder*. Пошто је у тој области спектар примјене и број захтијева доста већи тако имамо и већи број различитих врста чворова. Поред чворова који иницирају трансакције (клијената), *Hyperledger* је увео и неколико нових улога које на себе преузимају дио посла који је раније био обављан комплетно на једној врсти чворова. То су чворови који пружају услуге уређивања трансакција у блокове (енгл. *orderers*) и одобрења трансакција (енгл. *endorsers*) [8]. Улога *endorser* чворова јесте да изврше локално одређену трансакцију коју треба да одобре, утврде да ли је изводљива и валидна и у колико јесте својим потписом клијенту који је иницирао трансакцију врате потврду о одобрењу. Међутим, у великом броју случајева одобрење једног *endorser*-а не значи да је трансакција валидна и спремна за уписивање у *blockchain*. Некада је за уписивање трансакције потребно више од једног одобрења од стране *endorser*-а. Број одобрења који је потребан да би се трансакција сматрала валидном и била уписана у *blockchain* специфициран је у политици одобрења (енгл. *endorsment policy*) која се разликује од трансакције до трансакције.

Чворови који уређују трансакције, такозвани orderer чворови, прижају на мрежи канале комуникације између клијената и осалих чворова који одржавају *blockchain* базу и дневник трансакција. Клијети користе услуге orderer чворова који емитују трансакције и достављају их осталим чворовима. Дати канали комуникацију

обезбјеђују да ће иста порука, односно трансакција, бити достављена сваком повезаном чвору и то у истом редослиједу.

2.1.3. Типови мрежа

Као што у различитим имплементацијама постоје различити типови чворова тако постоје и различити типови мрежа. Неке мреже су дизајниране тако да буду потпуно отворене, децентрализоване и анонимне. Било ко може да се прикључи на мрежу и одлучује о томе које трансакције су валидне у могу ући у *blockchain* базу а које не. Такве мреже су јавне, попут интернета, доступне су свима и све што је потребно за учешће јесте утрошак компјутерских ресурса да би се мрежа заштитила од напада. Управо због ових особина, такве мреже се називају јавне мреже (енгл. *permissionless*). Због своје природе која захтијева рјешавање бројних задатака и утрошак ресурса, ове мреже су скупе за одржавање. Са друге стране постоје и мреже са правима приступа (енгл. *permissioned*) које у својој основи воде рачуна о учесницима. Такве мреже су приватне са ограниченим правима приступа, као што је примјер са интранетом. Пошто је за учешће у одлукама на мрежи потребан дозвола и утврђен идентитет нема потребе за утрошком ресурса те су захваљујући тој особини и скривености података ове мреже доста погоднije за индустријску примјену. У табели 1 приказане су особине јавних мрежа и мрежа са правима приступа [9].

	Мреже са правима приступа	**Јавне мреже**
1) Ко одржава мрежу?	Један или мала група одабраних чланова са додјељеним правима приступа	Сви
2) Како су подстакнути да постпају исправно?	Одржавање репутације	Економски подстицај, награде у виду токена
3) Ко креира податке (може да иницира трансакције)?	Група људи са додјељеним правима	Сви
4) Ко има право приступа бази?	Један или мала група одабраних чланова са додјељеним правима приступа	Сви
5) Гдје се чува база?	Централни сервери	Дистрибуирано
Потреба за централном фигуром која чува податке?	+	-
Трошкови одржавања дневника трансакција	-	+
Брзина	-	+
Отпорност на цензуру	Не	Да
Потреба за токенима	Не	Да
Представници	Hyperledger	Bitcoin, Ethereum

Табела 1. Особине јавних мрежа и мрежа са правима приступа

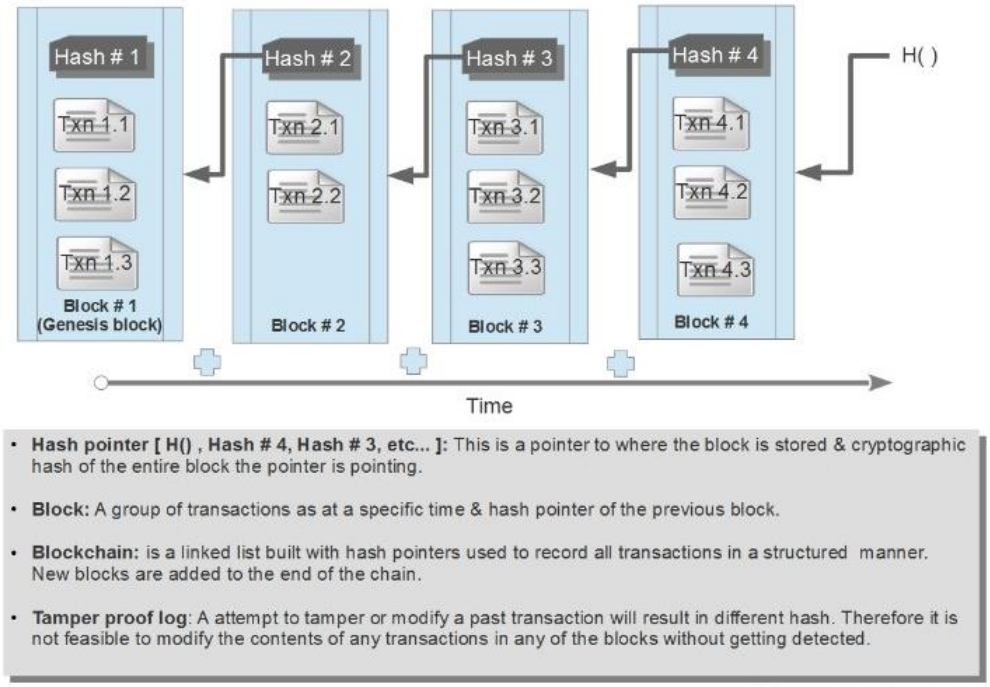
2.2. Структура отпорна на измјене

Једна од главних особина *blockchain* база јесте њихова отпорност на измјене. Све што се једном упише у ланац остаје ту заувијек, не постоји могућност брисања или измјене података. То је постигнуто захваљујући хеш показивачима.

Као што је већ поменуто, *blockchain* назив проистиче из структуре која је организована у форми ланца блокова. Сваки блок у ланцу повезан је са претходним преко хеш показивача. Хеш показивач не представља ништа друго осим обичног показивача који указује на локацију одређеног податка, уз који се чува и хеш вриједност тог истог податка на који показује. Док обични показивачи служе за добављање података са одређене локације, хеш показивачи нуде и могућност провјере да подаци нису мијењани од тренутка настанка показивача. На слици 3 приказан је хеш показивач [3].

Слика 3. Хеш показивач

Захваљујући хеш показивачима, од познатих структура података можемо лако добити нове са додатним особинама. Баш то је и урађено када је у питању *blockchain*. *Blockchain* ланац представља једноструко спрегнуту листу у којој су обични показивачи замијењени хеш показивачима. Сваки блок у ланцу поред тога што нам говори гдје се налази претходни, такође садржи и хеш садржаја претходног блока који нам омогућава да лако провјеримо да ли су подаци унутар блока мијењани или не. На слици 4 приказана је структура *blockchain* ланца [10].

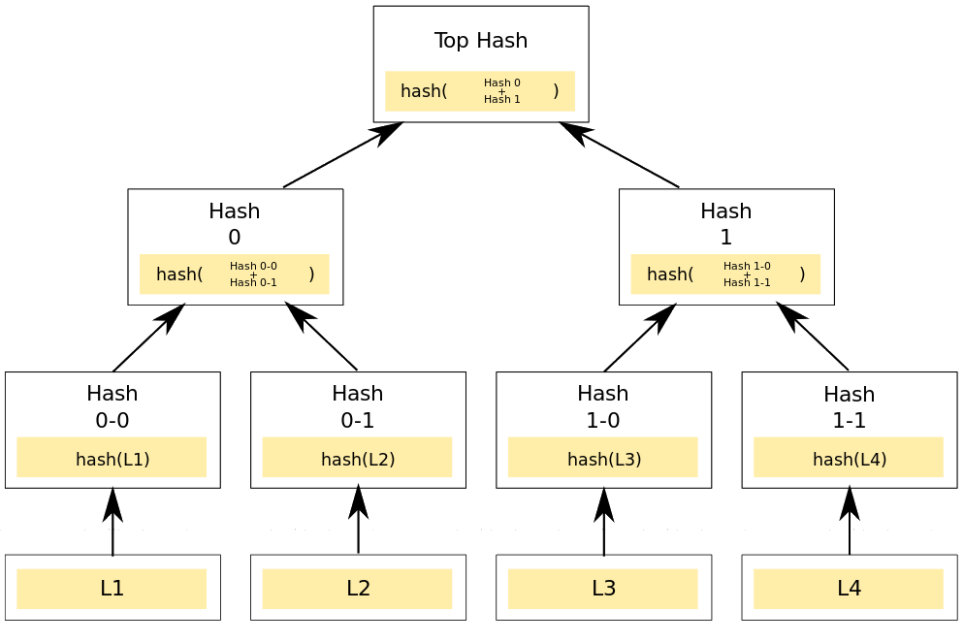


Слика 4. Структура blockchain ланца

У колико неко покуша да промијени податке било гдје у ланцу, то ће бити детектовано. Рецимо, на примјер, да је измијењена трансакција у неком блоку *k* који се налази негдје у средини ланца. Пошто је садржај блока *k* измијењен онда ће и хеш вриједност читавог блока да се промијени. Захваљујући статистици и особинама хеш функција можемо са веома великом сигурношћу да трвдимо да ће промјена вриједности садржаја блока довести до промјене хеш вриједности. У том случају хеш показивач из блока *k* + 1 неће више показивати на тај блок као што је био случај раније и на тај начин смо детектовали измјену која се десила у блоку *k*. Да би се уклонило неслагање између показивача из блока *k* + 1 и садржаја блока *k* довољно је замијенити дати показивач новим. Та измјена међутим, опет представља измјену садржаја блока, у овом случају блока *k* + 1, те се опет јавља проблем са показивачем у наредном блоку. Наравно, могуће је наставити мијењати показиваче све док се не дође до краја ланца али уколико се хеш показивач са краја ланца чува на сигурном мјесту на којем му нико не може приступити и измијенити га, моћи ћемо примјетити измјену која се деси било гдје у ланцу. Дакле, памћењем само једне хеш вриједности са краја ланца заправо се гарантује да ће читава структура *blockchain* базе бити отпорна на измјене, без обзира на број блокова који се налази у ланцу.

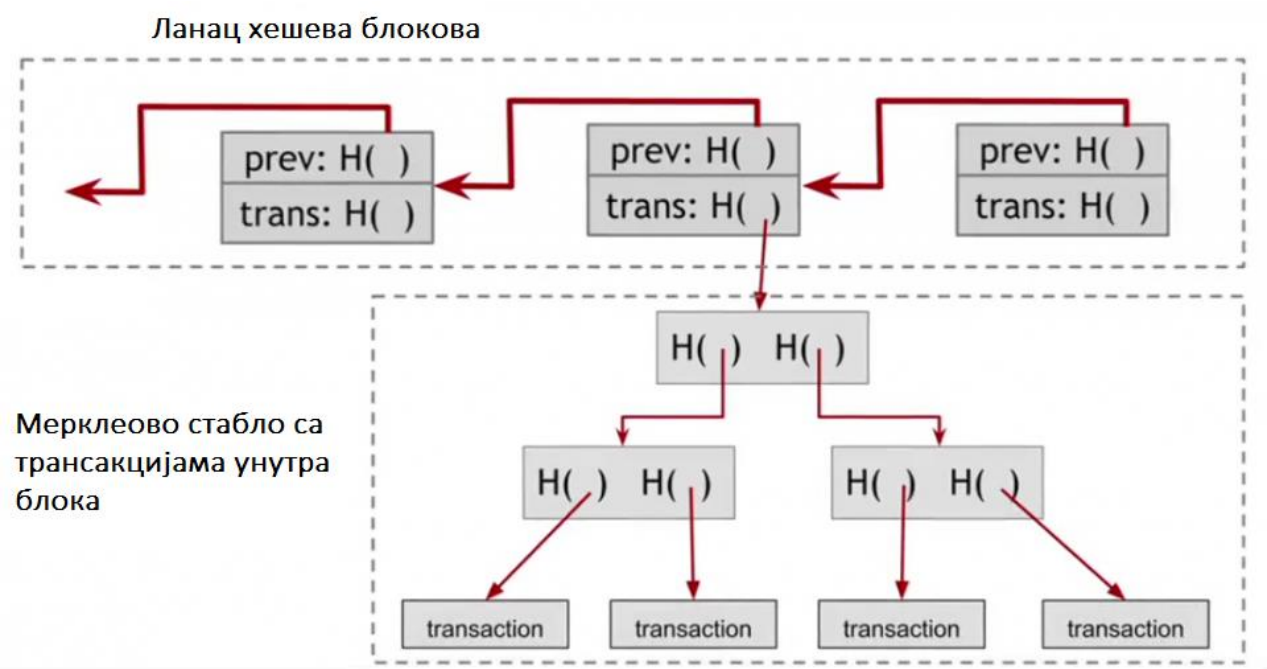
Поред спрегнутих листа, хеш показивачи су се такође показали корисни и када су у питању бинарна стабла. Бинарно стабло у коме су обични показивачи замијењени хеш показивачима назива се Мерклеово стабло (енгл. *Merkle tree*). У Мерклеовим стаблима листови представљају податке. Листови се групишу у парове по два и за сваки пар формира се структура која садржи два хеш показивача, један за сваки од листова. Ове структуре чине следећи ниво Мерклеовог стабла. Исти поступак се понавља за сваки следећи ниво све док се не стигне до једног чвора који представља

коријен стабла. Управо оваква структура је искоришћена за организовање и чување трансакција унутар блока. На слици 5 приказана је структура Мерклеовог стабла [11].



Слика 5. Структура Мерклеовог стабла

Као и код листа довољно је да запамтимо само хеш показивач са врха стабла и на основу њега обезбијеђен нам је пролазак низ читаво стабло до било које трансакције. Уједно тај исти показивач гарантује нам да се подаци унутар стабла нису мијењали јер би се, као што је већ објашњено на примјеру *blockchain*-а, свака измјена пропагирала на ниво изнад и евентуално дошла до показивача у коријену стабла. Све док је показивач на корјен стабла запамћен на безбиједном мјесту не постоји могућност измјене података а да се то не детектује. На слици 6 приказано је на који начин су унутар *blockchain*-а примјењене поменуте структуре података (једноструко спрегнуте хеш листе и Мерклеова стабла) [3].



Слика 6. Трансакције организоване у форми Мерклвог стабла унутар blockchain-а

Захваљујући Мерклеовим стаблима довољно запамтити само један хеш показивач умјесто читавог стабла трансакција. Пошто је количина података коју у том случају требамо запамтити доста мања, она се може смјестити и у заглавље блока заједно са хеш показивачем на претходни блок. У том случају да би се очувала претходно поменута отпорност комплетног ланца на измјене, умјесто читавог садржаја блока, довољно је узети само хеш вриједност заглавља приликом повезивања блокова у ланац

Још једна корисна особина коју Мерклеова стабла доносе јесте лакоћа утврђивања постојања трансакције у блоку. Да би се показало да се одређена трансакција налази у блоку довољно је показати пут, односно чворове који се налазе између коријена стабла и дате трансакције. Преко хеш показивача који се налазе у чворовима веома је лако доћи трансакције коју тражимо, без претраживања остатка стабла. Ако се на примјер, у стаблу налази n трансакција у том случају за потврђивање постојања трансакције потребно је приказати само око *log(n)* чворова, што знатно скраћује утрошено вријеме чак иако у стаблу постоји велики број трансакција [3].

Поред утврђивања постојања трансакција, Мерклеова стабла су корисна и када је у питању утврђивање да нека трансакција није члан одређеног блока. У ту сврху користи се сортирано Мерклеово стабло. То је стабло у ком се листови сортирају у одређеном редослиједу, који може бити алфабетски, нумерички или добијен на неку други начин. Да би се показало се тражена трансакција не налази у стаблу на које показује хеш показивач датог блока, довољно је показати путеве до трансакција која би требале да се налазе испред и иза дате трансакције која се тражи. У колико та два пута воде до узастопних листова у стаблу, а знамо да је стабло сортирано, показали смо да дата трансакције није члан тог стабла. Као и у претходном случају са потврђивањем

постојања, вријеме које је потребно да се докаже да трансакција није у датом блоку пропорционално је логаритму укупног броја трансакција у стаблу [3].

2.3. Консензус

Кључни услов који треба бити задовољен да би *blockchain* технологија функционисала јесте да се сви чланови мреже сложе око садржаја базе. Садржај *blockchain* базе се модификује преко консензус протокола који осигурава опште прихваћен редослијед трансакција и блокова у ланцу и гарантује интегритет и конзистенцију садржаја код свих чланова мреже.

Blockchain мреже се могу подијелити у двије врсте, јавне мреже и мреже са правима приступа (енгл. *permissioned* и *permissionless*). Код јавних мрежа, које су отворене за коришћење, сваки члан мреже има могућност извршавања трансакција и учествовања у консензусу. Са друге стране, код мрежа са правима приступа, право изварашавања трансакција има сваки члан док је учествовање у консензусу ограничено на одређен скуп чланова са утврђеним идентитетима. Пошто се у јавним мрежама очекује велики број анонимних и непоузданих чланова, механизам за постизање консензуса треба да буде отпоран на злоупотребе, а посебно такозване Сибил нападе (енгл. *Sybil attacks*). Сибил напади су карактеристични за *peer-to-peer* мреже и представљају покушај појединаца да генеришу више лажних идентитета и на тај начин већинским присуством угрозе или контролишу процес консензуса [12]. Код *Bitcoin*-а, као представника јавних мрежа, тај проблем је ријешен пројектовањем консензуса који се назива доказ посла и који захтијева рјешавање задатка захтјевног за израчунавање. Да би учествовали у консезусу и добили право додавања блокова у ланац, чланови морају да потроше значајну количину компјутерских ресурса и енергије. Такав приступ обесхрабрује и чини Сибил нападе јако скупим, те је неопходан за сигурност процеса консензуса и читаве *blockchain* базе.

Мреже са правима приступа имају дјелимично поуздане чланове, односно дио чланова мреже који имају права уписа података у *blockchain* базу су провјерени и поуздани. Захваљујући томе овакве врсте мрежа не захтијевају потрошњу велике количине енергије зарад сигурности и могу примјенити другачије механизме за постизање консензуса као што је Византијска толеранција грешке.

Централну улогу у избору *blockchain* платформе игра сигурност модела консензуса. Имплементација лошег алгоритма за постизање консензуса може компромитовати податке у *blockchain* ланцу и онемогућити испуњење функционалних захтијева технологије.

Постизање консензуса у дистрибуираним системима је изазован подухват. Добар алгоритам мора бити отпоран на привремену недоступност чворова, кашњење и корупцију порука, као и злонамјерне чланове. Алгоритам за постизање консензуса посједује три особине на основу којих се одређује његова ефикасност и то су [13]:

- Сигурност – Алгоритам се сматра сигурним у колико сви чланови који поштују правила протокола донесу исту одлуку, односно сложе са

тренутним стањем базе. Ова особина се такође може представити и као конзистентност садржаја базе.

- Животност – Алгоритам посједује особину животности у колико сви поштени чланови евентуално донесу одлуку.
- Толеранција на грешке – Алгоритам посједује толеранцију на грешке у колико може исправно да функционише чак и у случају губитка члана који учествује у консензусу.

У наставку овог поглавља описано је неколико алгоритама за постизање консензуса који се користе у различитим имплементацијама *blockchain*-а.

2.3.1. Практична византијска толеранција грешке (енгл. *Practical Byzantine Fault Tolerance*)

Практична византијска толеранција грешке је алгоритам који је настао као рјешење проблема који је карактеристичан за дистрибуиране системе, познатог под називом „Проблем византијских генерала“ (енгл. *Byzantine Generals Problem*). Проблем је представљен у виду приче у којој више дивизија византијске војске треба да нападне исти град. Сваком од дивизија управља по један генерал и генерали међусобно комуницирају путем гласника да би се договорили да ли да се напад изврши или не. Међутим, неки од генерала могу бити издајници који покушавају да спријече одане генерале да дођу до споразума. Да би напад био успјешан већина генерала треба да донесе одлуку да нападне град. Циљ овог проблема јесте да се осигура да ће се сви одани генерали донијети исту одлуку и да мали број издајника неће бити у стању да их доведе до доношења погрешне одлуке. Доказано је да је то могуће само уколико су једна трећина генерала, или мање, издајници [13].

У случају *blockchain* дистрибуираних база, чланови мреже представљају генерале, гласници су средства комуникације унутар мреже а одлука о нападу је одлука о томе да ли да се нови податак који се додаје у базу прихвати као валидан или не. Одани генерали из дате приче могу се представити као чланови мреже који поштују задата правила и имају у циљу очување интегритета базе, а издајници би били они чланови који за циљ имају подметање искварених података.

Алгоритам функционише тако што сваки члан, односно генерал, након пријема поруке, на основу одређених провјера сам доноси одлуку о томе да ли је порука валидна или не. У колико донесе одлуку да је порука валидна, дати члан ће то потврдити својим дигиталним потписом. Ако порука добије довољан број потписа биће сматрана валидном и уписана у *blockchain*. На тај начин се постиже консензус. Овај начин постизања консензуса захтијева мањи утрошак ресурса и енергије али није примјенљив у системима који подржавају анонимност и слободан приступ мрежи.

2.3.2. Доказ посла (енгл. *Proof of Work*)

Доказ посла алгоритам захтијева од чланова мреже да ријеше захтјевне криптографске задатке и на тај начин остваре право додавања блока у *blockchain*. Основни недостатак овог приступа јесте велика потрошња енергије, као и брзина

алгоритма. *Bitcoin* имплементација *blockchain*-а, која се користи доказ посла, подржава максимално седам трансакција по секунди. Новије имплементације *blockchain*-а имплементирају и новије алгоритме, који омогућавају бржи и ефикаснији избор кандидата [13].

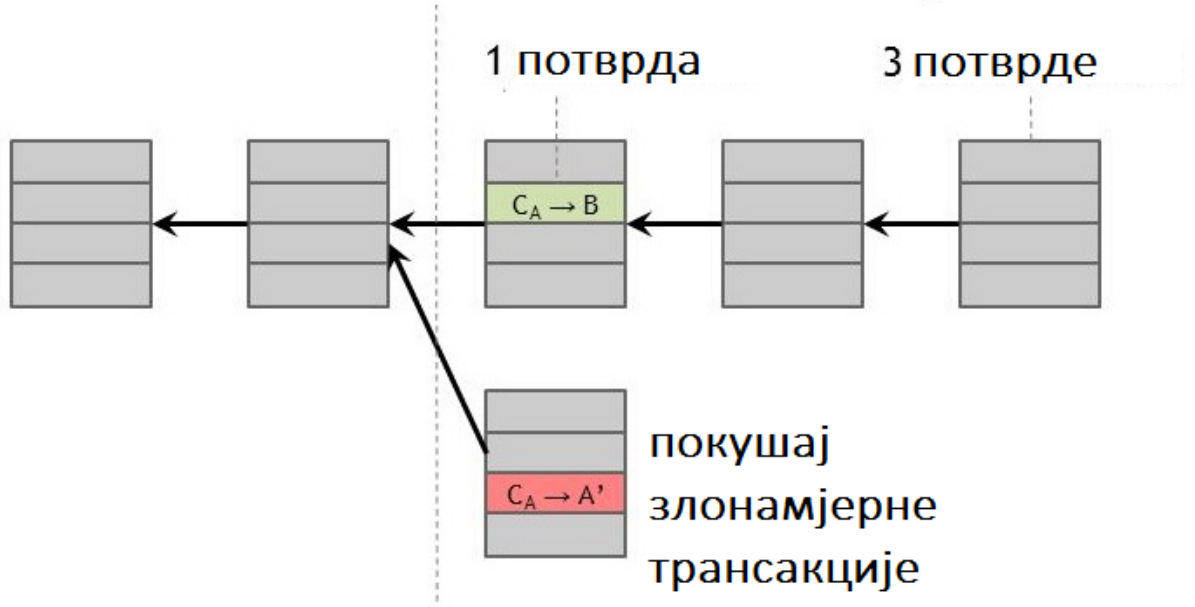
Доказ посла је први алгоритам за постизање дистрибуираног консензуса који се јавља у *Bitcoin* имплементацији *blockchain*-а. У датом алгоритму сви чланови мреже имају могућност одржавања конзистентности *blockchain* базе и такмиче се да ријеше задатак који захтијева коришћење хеш функција. Главни циљ задатка јесте да се пронађе случајан број (енг. *nonce*), који у комбинацији са заглављем блока, даје хеш вриједност одређених особина. Ако хеш вриједност не задовољава задате особине, бира се нови број и понавља поступак. Тренутна тежина задатка захтијева од чланова који се такмиче да испробају милијарде различитих бројева да би пронашли одговарајући. Особине хеш функција гарантују да ће мале разлике улаза проузроковати драстичне промјене излаза и да не постоји пречица која би омогућила лако налажење траженог броја. На слици 7 приказани су примјери промене улаза и одговарајућих излаза који се добију примјеном *SHA-256* хеш алгоритма [14]. Члан који први обезбиједи доказ извршеног посла, односно пронађе одговарајући број, добија право додавања наредног блока у ланац и награду у виду одређене количине токена, односно криптовалуте. Награда се састоји од посебне трансакције која ствара новац и додјељује га члану који је додао блок, као и од вишка новца у трансакцијама који представља нешто попут провизије.

```
I am Satoshi Nakamoto0 => a80a81401765c8eddee25df36728d732...
I am Satoshi Nakamoto1 => f7bc9a6304a4647bb41241a677b5345f...
I am Satoshi Nakamoto2 => ea758a8134b115298a1583ffb80ae629...
I am Satoshi Nakamoto3 => bfa9779618ff072c903d773de30c99bd...
I am Satoshi Nakamoto4 => bce8564de9a83c18c31944a66bde992f...
I am Satoshi Nakamoto5 => eb362c3cf3479be0a97a20163589038e...
I am Satoshi Nakamoto6 => 4a2fd48e3be420d0d28e202360cfbaba...
I am Satoshi Nakamoto7 => 790b5a1349a5f2b909bf74d0d166b17a...
I am Satoshi Nakamoto8 => 702c45e5b15aa54b625d68dd947f1597...
I am Satoshi Nakamoto9 => 7007cf7dd40f5e933cd89fff5b791ff0...
I am Satoshi Nakamoto10 => c2f38c81992f4614206a21537bd634a...
I am Satoshi Nakamoto11 => 7045da6ed8a914690f087690e1e8d66...
I am Satoshi Nakamoto12 => 60f01db30c1a0d4cbce2b4b22e88b9b...
I am Satoshi Nakamoto13 => 0ebc56d59a34f5082aaef3d66b37a66...
I am Satoshi Nakamoto14 => 27ead1ca85da66981fd9da01a8c6816...
I am Satoshi Nakamoto15 => 394809fb809c5f83ce97ab554a2812c...
I am Satoshi Nakamoto16 => 8fa4992219df33f50834465d3047429...
I am Satoshi Nakamoto17 => dca9b8b4f8d8e1521fa4eaa46f4f0cd...
I am Satoshi Nakamoto18 => 9989a401b2a3a318b01e9ca9a22b0f3...
I am Satoshi Nakamoto19 => cda56022ecb5b67b2bc93a2d764e75f...
```

Слика 7. Мале разлике улазних стрингова узрокују драстичне промјене излаза код SHA-256 алгоритма

Доказ посла не захтијева гласање свих чланова мреже да би се постигао консензус. Он се заснива на чињеници да је јако захтјевно и скупо додати нови блок у *blockchain*, и да блокови могу бити искључени из главног (најдужег) ланца. Захваљујући транспарентности података врло је једноставно провјерити валидност блока, те чланови врше одговарајуће провјере чак и након што су блокови додати у *blockchain*. Приликом

додавања блока провјерава се валидност последњих блокова који се налазе на најдужем ланцу и нови блокови се по правилу додају на последњи блок који се сматра валидним, односно који је успјешно прошао све провјере. То значи да сваки новододати блок повећава број потврда валидности блокова који се налазе иза њега. Ако неки од чланова покуша да дода блок у ком се налазе злонамјерне трансакције или трансакције које не поштују правила система, тај блок ће бити занемарен од стране поштених чланова који се неће наставити надовезивати на њега. На слици 8 приказана је илустрација главног ланца и занемарених блокова. Занемарени блокови не добијају награду за рјешавање задатка.



Слика 8. Примјер занемареног блока ван главног ланца

Једини случај када злонамјерни члан мреже може бити сигуран да његов блок неће бити искључен из ланце јесте у колико посједује огромну процесорску моћ и количину ресурса који би му омогућили да сваки пут побједи у рјешавању криптографског задатка. То значи да дати члан мора да посједује више од половине укупне процесорске моћи мреже те се такав напад назива „51% напад“ (енгл. *51% attack*). Сматра се да је вјероватноћа такавог напада изразито мала јер захтијева огромну количину ресурса, како хардверских тако и енергетских, те се долази до закључка да је члановима мреже исплативије да се понашају поштено [15][16].

2.3.3. Доказ посједовања (енгл. *Proof of Stake*)

Због велике количине процесорске моћи која је неопходна, доказ посла троши много енергије што га чини скупим. Из тог разлога људи су се почели окретати ка алгоритма који нуде исте функционалности а не захтијеавају толики утрошак ресурса. Најпознатија замјена за доказ посла јесте доказ посједовања. У овом алгоритму, за постизање консензуса, није потребно уложити у скупе хардверске компоненте и

енергију него чланови мреже улажу дио свог капитала у сам систем. Код доказа посједовања, шансе да одређени члан буде изабран за додавање новог блока зависе од количине новца који тај члан посједује у систему. Члан са 300 новчића има три пута веће шансе да буде изабран, него члан који посједује 100. Као што је био сличај и код доказа посла, да би члан који жели да нападне и угрози мрежу успјешно извршио напад потребно је да посједује 51% укупног новца у систему, што би било јако тешко извести.

Једна од разлика која се издваја у односу на доказ посла јесте да не постоји могућност стварања новца. Количина новца која се налази у систему је константна од настанка система и не мијења се. То значи да се награда у системима који користе доказ посједовања састоји само од провизија, не постоји посебна трансакција која датом члану додјељује новостворени новац [15]. Из тог разлога, и због чињенице да не постоји потреба за улагањем у хардверску опрему, доказ посједовања се сусреће са проблемом који се назива „Ништа на коцки“ (енгл. *Nothing-at-Stake*). Због поменуте природе награде чланови немају исти подстицај да се понашају поштено као што је случај са доказом посла, а пошто претходно нису уложили новац у опрему, немају шта ни да изгубе. Један од начина да се ријеши овај проблем предложила је *Ethereum* платформа, а то је варијација доказа посједовања коју су назвали Каспер. У Каспер алгоритму, сваки непоштен покушај додавања новог блока кажњава се истом количином новца коју би члан добио да је генерисао валидан блок.

Још један проблем са којим се сусреће доказ посједовања јесте чињеница да алгоритам, додјељивањем награде оним члановима који већ посједују већи проценат капитала у систему, ствара све више централизован систем.

У табели 2 упоређене су особине поменутих алгоритама за постизање консензуса [13].

	Доказ посла	Доказ посједовања	Византијска толеранција грешке
Тип blockchain мреже	Јавне	Јавне и мреже са правима приступа	Са правима приступа
Потврда трансакција	Одложена	Одложена	Тренутна
Брзина трансакција	Мала	Велика	Велика
Скалабилност мреже	Висока	Висока	Ниска
Цијена учешћа	Да	Да	Не
Отпорност на нападе	<=25%	Зависи од алгоритма	<=33%
Повјерење	Непоуздани	Непоуздани	Дјелимично поуздани

Табела 2. Поређење најпознатијих алгоритама за постизање консензуса

Тип *blockchain* мреже – Односи се на тип *blockchain* мрежа за које је алгоритам осмишљен. Иако је доказ посла првенствено намјењен јавним мрежама, може се примјенити и у мрежама са правима приступа али у том окружењу неће имати исте перформансе.

Потврда трансакција – Односи се на коначност потврде трансакција. У доказ посла и доказ посједовања алгоритмима није гарантовано да ће трансакције које уђу у *blockchain* остати укључене у ланцу јер постоји шанса одбацивања блокова. То доводи до чињенице да клијети морају чекати дуже да би били сигурни да ће трансакција остати прихваћена и валидна.

Брзина трансакција – Односи се на број трансакција по секунди. Алгоритми који могу бржи потврдити трансакције и гарантовати њихову валидност имају и већи проток трансакција. Доказ посла подразумијева компјутерски и временски захтјевна рачунања као и дуже чекање на потврду те је број трансакција које може да обради знатно мањи него код алгоритама као што је византијска толеранција грешке.

Скалабилност мреже – Представља способност алгоритма да постигне консензус под условом да број чланова константно расте. Доказ посла и доказ посједовања су првенствено намјењени отвореним мрежама гдје се очекује велики број чланова и гарантује анонимност те имају много већи степен скалабилности у односу на винзантијску толеранцију грешке. За византијску толеранцију грешке препоручује се да број чланова који учествују у постисању консензуса буде око 20. Повећање тог број изнад 20 доводи до повећања броја порука које треба да се размијене између чланова те што резултује већим трошковима.

Цијена учешћа – Односи се на цијену учешћу у постизању консензуса. Код доказа посла то су ресурси у виду енергије и процесорске моћи а код доказа посједовања у виду капитала, односно новца у систему.

Отпорност на нападе – Представља проценат мреже који може бити компромитован без угрожавања рада алгоритма.

Повјерење – Односи се на анонимност чланова. Код система који користе доказ посла и доказ посједовања сви чланови су анонимни и имају право учешћа у постизању консензуса јер се алгоритми ослањају на друге механизме када је у питању сигурност. Са друге стране, у системима који користе византијску толеранцију грешке чланови који учествују у постизању консензуса морају имати унапријед утврђене и провјерене идентитете.

2.4. Паметни уговори

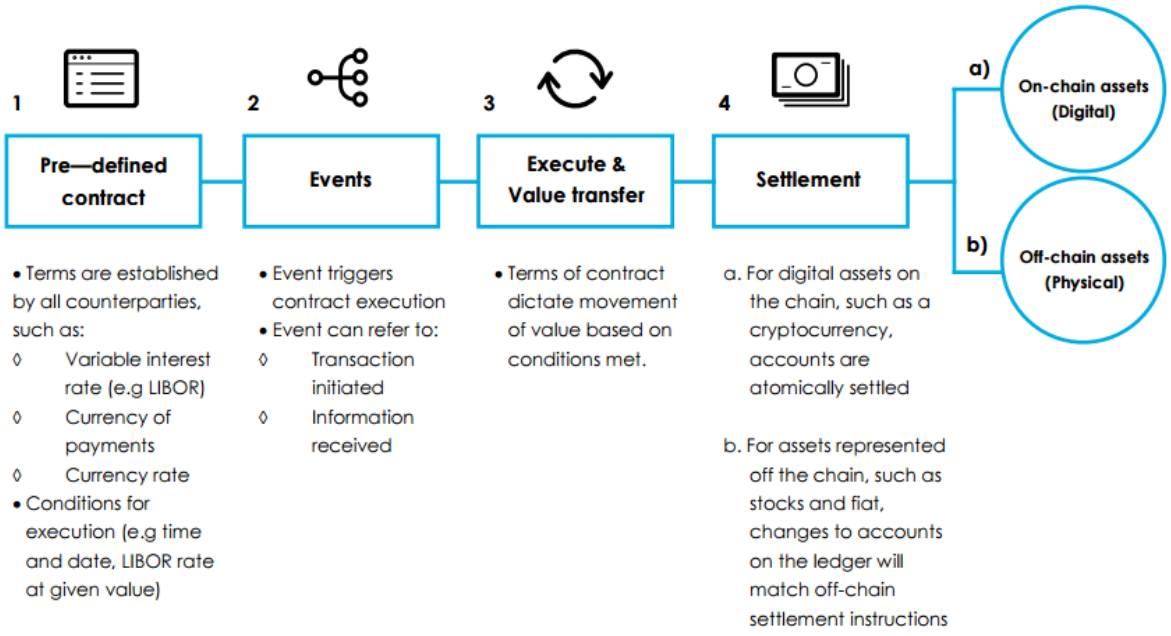
У пословној терминологији појам уговор подразумијева договор између двије или више странака о размјени добара, услуга или информација. У уговору се јасно дефинишу правила понашања и одговорности странака, начини на који се одређене акције морају спроводити, последице до којих може доћи у колико нису задовољени услови дефинисани уговором и још много тога.

Сва горе наведена правила могу се кодирати у неком програмском језику. Да би се такав уговор могао сматрати паметним уговором (енгл. *smart contract*) потребно је да посједује следеће особине [17]:

- уговор мора да садржи макар једну операцију која је комплекснија од операције преноса средстава (криптовалуте) између двије странке,
- уговор мора да укључује двије или више странака,
- када уговор активира, операције дефинисане уговором морају се извршавати без људског утицаја.

Захваљујући последњој особини ови уговори су добили назив паметни уговори. *Blockchain* технологија представља одличну основу за имплементацију паметних уговора. Концепт паметних уговора први је предложио Ник Сабо (енгл. *Nick Szabo*) 1996. године [18], а *Ethereum* је прва платформа која је укључила тај појам у своју имплементацију. Паметни уговори представљају договор, или скуп правила који усмјеравају пословне трансакције. У случају испуњења договорених услова, уговор се сам активира и извршава операције договорене приликом креирања уговора. Једна од највећих предности паметних уговора јесте да обезбјеђују финалан скуп исхода који

гарантују да никада неће доћи до забуна и неће постојити потреба за судским поступцима. На слици 9 приказан је ток извршавања паметног уговора [19].



Слика 9. Ток извршавања паметних уговора

3. *HYPERLEDGER FABRIC*

У овом поглављу биће ријечи о *Hyperledger* пројекту и *Fabric*-у који представља једну од имплементација *blockchain* технологије и развија се као посебан пројекат унутар *Hyperledger*-а. Последњи дио овог поглавља посвећен је поређењу *Hyperledger Fabric*-а и још једне од познатијих имплеметација *blockchain*-а, Ethereum-а.

3.1. Пројекат *Hyperledger*

Пројекат *Hyperledger* је *open source* сарадничка иницијатива створена са циљем унапријеђивања *blockchain* технологија који би спојиле различите секторе индустрије [20]. Пројекат је вођен од стане *Linux* фондације и у њему учествују водећи представници из области финансија, банкарства, производње и технологије. Главни циљ јесте да се развију *blockchain* мреже и дистрибуирани дневници трансакција који би пружили могућност подржавања глобалних пословних трансакција које би се одвијале између система из различитих области индустрије и унаприједили перформансе и поузданост тих система.

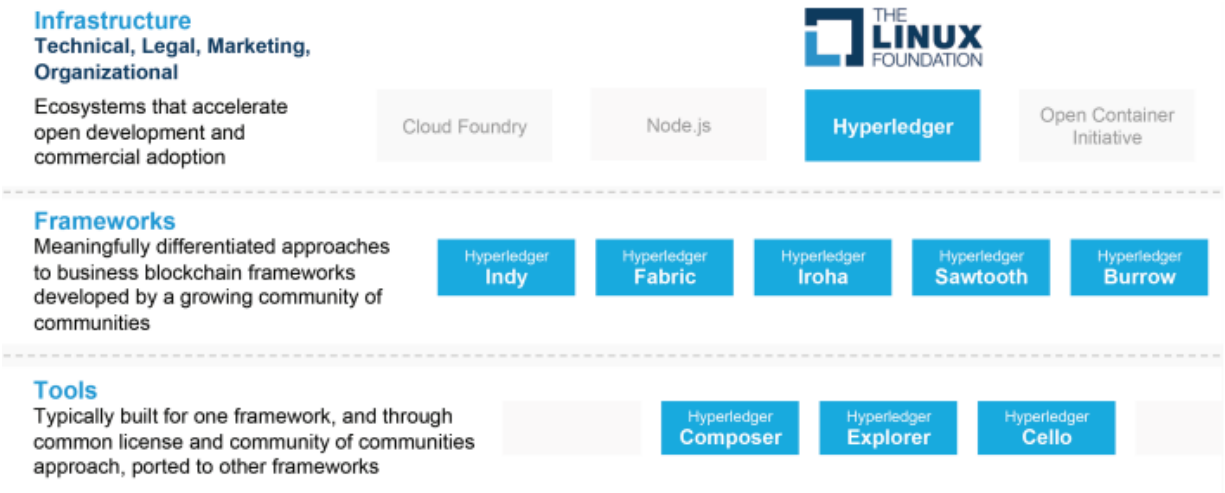
У *Hyperledger* пројекту за сада учествује преко 100 чланова од којих се издвајају гиганти као што су *Airbus*, *Daimler*, *IBM*, *Fujitsu*, *Intel*, *Samsung*, *American Express* и многи други [20].

Под окриљем *Hyperledger* пројекта покренути су и други пројекти који су усмјерени ка развоји разних *blockchain* технологија, радних окружења, библиотека, интерфејса и алата. Пројекти које је започео *Hyperledger* су [20]:

- ***Hyperledger Sawtooth***: Модуларна платформа за развој дистрибуираних трансакционих днеквника, вођена од стране *Intel*-а, која користи нови алгоритам за постизање консензуса који се назива Доказ протеклог времена (енгл. *Proof of Elapsed Time*).
- ***Hyperledger Iroha***: Пројекат који има за циљ развој пословне *blockchain* платформе коју ће се моћи лако и једноставно укључити у инфраструктуралне пројекте.
- ***Hyperledger Fabric***: Пројекат вођен од стране *IBM*-а. *Hyperledger Fabric* је имплементација *blockchain* технологије са модуларним компонентама као што су алгоритам за постизање консензуса и сервиси за чланства.
- ***Hyperledger Burrow***: *Hyperledger Burrow* је модуларан *blockchain* клијент са интерпретером паметних уговора уграђеним унутар спецификације *Ethereum* виртуалне машине (*EVM*).
- ***Hyperledger Indy***: Нуди алате, библиотеке и поновно искористиве компоненте за обезбјеђивање дигиталних идентитета.

- ***Hyperledger Cello*:** Пројекат који има за циљ смањење напора који су потребни за креирање, одржавање и гашење *blockchain* мрежа.
- ***Hyperledger Composer*:** *Composer* је алат намјењен за развој *blockchain* пословних мрежа. Главни циљ му је убрзање процеса развоја паметних уговора.
- ***Hyperledger Explorer*:** Пружа услуге читања, позивања и претраживања блокова, трансакција, паметних уговора и осталих информација везаних за мрежу.

На слици 10 прикани су пројекти унутар *Hyperledger*-а [21].



Слика 10. Пројекти унутар Hyperledger-а

Филозофија *Hyperledger* пројекта подразумијева развој модуларних, компатибилних компоненти и интерфејса који су лаки за коришћење и сви пројекти који се развијају под окриљем *Hyperledger*-а придржавају се поменуте филозофије. *Hyperledger Architecture* радна група дефинисала је следеће компоненте као дијелове пословних *blockchain* мрежа [21]:

- Слој консензуса – задужен је за постизање консензуса када су у питању редослијед и валидност трансакција које чине блокове.
- Слој паметних уговора – задужен је за обраду трансакционих захтијева и спровођење пословне логике и правила.
- Слој комуникације – задужен је за пренос порука између чланова мреже.
- Слој базе података – пружа могућност коришћења различитих база података.
- Слој криптографских алгоритама – пружа могућност лаке замјене криптографских алгоритама без ометања рада других компоненти.

- Сервиси за пружање идентитета – омогућавају успостављање централног мјеста повјерења, регистрацију нових идентитета као и операције аутентификације и дефинисања права приступа.
- Сервиси за дефинисање политика – задужени за дефинисање и одржавање разних политика везаних за систем, као што је политика одобравања трансакција (енгл. *Endorsement policy*). Ослања се на друге компоненте када је у питању спровођење датих политика.
- Апликативни програмски интерфејси – омогућавају комуникацију између клијентских апликација и *blockchain*-а.
- Компатибилност – обезбјеђује компатибилност између различитих *blockchain* инстанци.

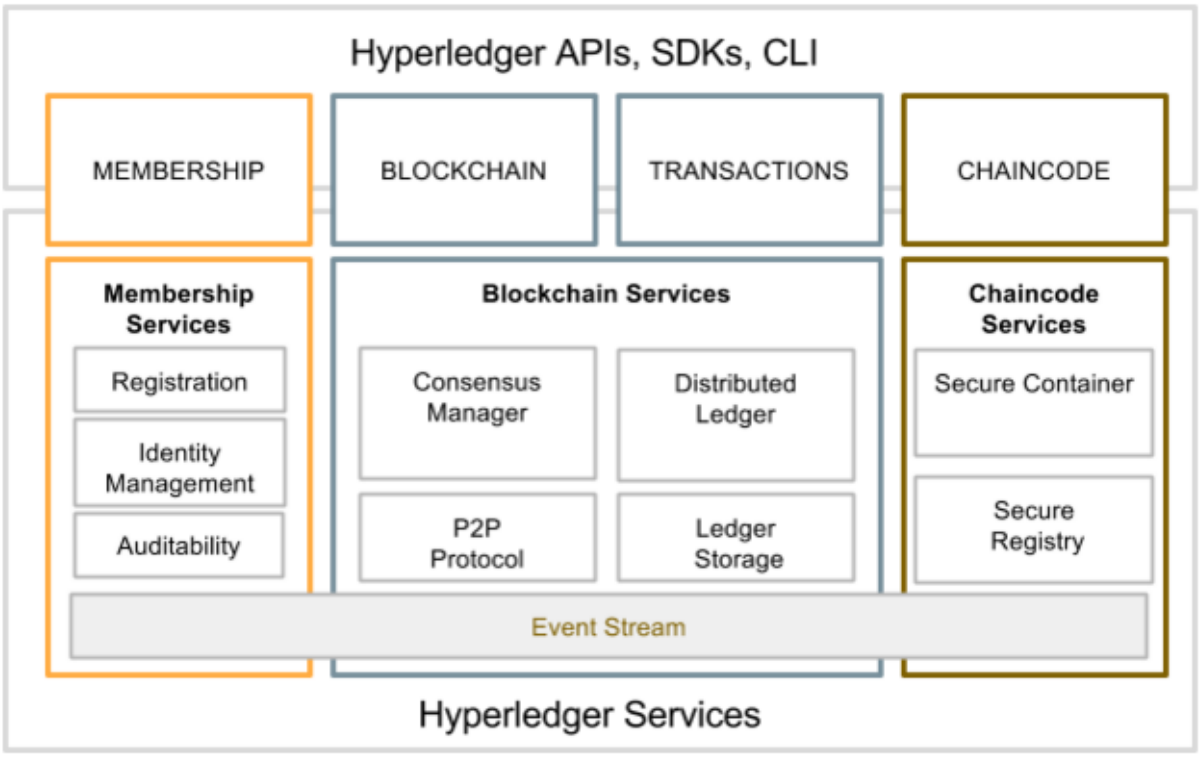
3.2. *Hyperledger Fabric*

Са појавом *Bitcoin*-а у свијету финансија 2009. године, многе компаније су почеле да се интересују и истражују технологију на којој се заснива популарна криптовалута. Доступност података омогућава креирање економичних пословних мрежа које не зависе од ценалног мјеста повјерења, насупрот трандиционалном приступу гдје сваки члан заједнице одржава сопствену базу података што захтијева усаглашавање трансакција на скуп, временски захтијеван и неефикасан начин. Поред тога, појава паметних уговора омогућила је да се пословна логика и правила аутоматизују, кодирају и смјесте на мрежу и тако постану јавно провјерљива и доступна свим члановима заједнице. Међутим, прве имплементације *blockchain* технлогије нису испуњавале одређене захтјеве који су неопходни када су у питању пословне трансакције. Проблем скалабилности, недостатак подршке за приватне и повјерљиве трансакције, анонимност учесника у мрежи и још много тога чини *blockchain* технологију неупотребљиву у развоју пословних апликација.

Да би задовољили поменуте захтјеве и приближили *blockchain* технологију индустријској употреби *Hyperledger* је започео пројекат под називом *Fabric*. *Hyperledger Fabric* је *blockchain* технологија која као и остале има дијељени дневник трансакција и користи паметне уговоре, али у односу на остале технологије *Hyperledger Fabric* подржава мреже које су приватне и са правима приступа. То је постигнуто тако што су уведени сервиси чланства помоћу којих чланови добијају провјерене идентитете и приступ мрежи, што такође утиче и на перформансе система. Захваљујући идентитетима нема потребе за сигурносним механизмима који користе ресурсно захтјевне алгоритме као што је доказ посла. *Hyperledger Fabric* такође нуди модуларну архитектуру, подаци везани за мрежу могу се чувати у различитим форматима а механизми за постизање алгорима могу мијењати по портеби тако да на најбољи начин задовоље постављене захтјеве. Као што је већ поменуто, поред јавних мрежа *Hyperledger Fabric* подржава и мреже код којих је приватност и повјерљивост од кључног значаја. За разлику од информација на *blockchain*-у које су по правилу јавне и доступне свима, пословни уговори често захтијевају да се одређени дијелови и информације заштите и буду доступни само странкама које у укључене у уговор. То је нарочито битно за мреже у којима чланови

могу бити конкуренти и не желе своје пословне тајне да дијеле са осталим учесницима. Из тог разлога *Hyperledger Fabric* нуди могућност креирања посебних канала комуникације који имају сопствене дневнике трансакција. Поред канала, повјерљивост информација обезбјеђена је коришћењем криптографских метода које омогућавају да само чланови који учествују у трансакцији могу да декриптују и извршавају дате трансакције [22].

На слици 11 приказана је организација компоненти *Hyperledger Fabric*-а у три логичке цјелине [23].



Слика 11. Основни сервиси Hyperledger Fabric-а

3.2.1. Сервиси чланства

Улога сервиса чланства (енгл. *membership services*) јесте да врше аутентификацију и воде рачуна о идентитетима, приватности и правима приступа на мрежи. Сваки члан мреже мора да се региструје помоћу сервиса чланства да би добио идентитет и право извршавања трансакција. Модул сервиса чланства издаје дигиталне сертификате провјереним члановима који желе да учествују у мрежи, не извршава никакве трансакције нити има увид у употребу издатих сертификата.

Сервиси чланства нуде три основне функционалности: регистрацију чланова (енгл. *registration*), учлањивање (енгл. *enrollment*) и издавање сертификата за извршавање трансакција. Да би се могао издати сертификат, члан мора бити регистрован и учлањен у мрежу. Операције регистрације могу вршити само они чланови чији идентитет има право регистровања нових чланова. Операција

регистрације као излаз даје токен који се касније користи приликом учлањивања. Током учлањивања сервиси чланства генеришу посебан сертификат на основу ког се издају сертификати за извршавање трансакција. Помоћи сертификата за извршавање трансакције чланови могу извршавати трансакције са гаранцијом да било које двије трансакције покренуте од стране истог члана не могу бити повезане.

Пружалац сервиса чланства (енгл. *Membership Services Provider - MSP*) је компонента која пружа апстракцију сервиса чланства. *MSP* апстрахује све криптографске механизме и протоколе који се користе приликом издавања сертификата и аутентификацију чланова. У *Fabric* верзији 0.6 *MSP* је био критична тачка система (енгл. *Single Point of Failure*) али у верзији 1.0 *blockchain* мреже могу имати више пружалаца сервиса чланства [24].

3.2.2. *Chaincode*

Паметни уговори у *Hyperledger Fabric*-у се називају *chaincode*-ови. *Chaincode* сервиси користе *Docker* за безбједно извшавање паметних уговора од стране валидирајућих чланова. *Hyperledger Fabric chaincode*-ови представљају средство комуникације између нивоа података и екстерних апликација које их позивају када желе да прочитају или упишу податке у базу. *Chaincode* може да комуницира и са дневником трансакција као и са базом која чува тренутно стање ресурса у систему. *Chaincode* трансакције су временски ограничене као што је случај са веб захтјевима или трансакцијама у базама података. Ако се трансакције не заврши у одређеном временском периоду биће пријављена грешка и промјене настале извршавањем трансакције неће бити примјење на тренутно стање ресурса. *Hyperledger Fabric* тренутно подржава *Go* и *Java* програмске језике за имплементацију *chaincode*-ова.

Сваки *chaincode* мора да импементира интерфејс који има три методе које се позивају по пријему трансакција. Те методе су *Init, Query* и *Invoke* [25].

Init се позива када се *chaincode* први пут смести на *blockchain*. Приликом смјештања *chaincode* се шаљу у виду трансакције и прослијеђује свим члановима мреже. Сваки члан по пријему *chaincode*-а креира нови *Docker* контејнер у који се уграђују *chaincode*. Након тога контерјнер се покреће и позива се *Init* метода. Као што само име наговјештава, *Init* метода се користи са све иницијализације које су потребне *chaincode*-у.

Query се позива сваки пут када се претражују подаци. *Chaincode* ће прочитати тренутно стање ресурса на *blockchain*-у и вратити информације кориснику чија је трансакција позвала *Query* методу. Трансакције које само претражују податке не памте се у дневнику трансакција.

Invoke се позива када је потребно извршавање конкретних *chaincode* метода које извршавају одређену пословну логику. Позиви *Invoke* методе чувају се као трансакције које се организују у блокове и смјештају на *blockchain*. Сваки пут када треба да се промијени стање неког ресурса у бази позива се *Invoke* метода. Структура методе је једноставна, у зависности од прослијеђених параметара биће позвана одговарајућа помоћна функција која представља пословну логику или враћена грешка.

3.2.3. Организација података

Подаци у *Hyperledger Fabric*-у организовани су тако да се чувају унутар двије различите компоненте: дневнику трансакција (енгл. *Transaction Log*) и бази која чува тренутно стање ресурса у систему (енг. *World State*). Таква структура названа је *Ledger*. Сваки учесник посједује копију *Ledger*-а за сваку *Hyperledger Fabric* мрежу чији је члан. Дневник трансакција чува све трансакције које су утицале на ресурсе и довеле до тренутног стања ресурса у систему. Дневник трансакција се може посматрати као историја измјена ресурса. Као подразумијевана база за чување вриједности и стања ресурса користи се *LevelDB* кључ-вриједност база података. *Fabric* такође подржава и *CouchDB* базу која нуди све могућности *LevelDB* базе али има и подршку за компликованије претраге података које не користе само вриједости кључева [26].

3.2.4. Консензус

Hyperledger Fabric је дизајниран тако да има модуларну архитектуру. Захваљујући томе алгоритми за постизање консензуса се могу мијењати тако да на најбољи начин задовоље захтјеве система. Консензус је у *Hyperledger Fabric*-у је подијељен у три фазе: одобравање, уређивање и валидација [].

- Одобравање трансакције је одређено политиком одобравања која се дефинише посебно за сваки *chaincode.*
- Фаза уређивања прихвата одобрене трансакције и уређује их у блокове након чега су спремне да се додају у дневник трансакција.
- Валидација прихвата уређене блокове и валидира трансакције прије додавања у дневник трансакција. Валидација се врши да би се осигурало да од момента одобравања до момента додавања у дневник трансакција није дошло до промјене стања коришћених ресурса.

Hyperledger Fabric подржава модуларност за све три фазе консензуса. Апликације, у зависности од постављених захтјева, могу користити различите имплементације механизама за одобравање, уређивање и валидацију.

Да би боље разумјели како се постиже консензус унутар *Hyperledger Fabric* мреже у наставку су описана три различита типа чворова и њихове улоге.

1. Клијент : иницира трансакције, шаље их на одобравање и уређивање у колико прођу процес одобравања.
2. *Peer*: чвор који прима блокове уређених трансакција, врши валидацију и одржава дневник трансакција и стање ресурса. *Peer* чворови могу имати и посебну улогу одобравања (енгл. *endorsers*) која је задужена за одобравање трансакција прије уписивања у дневник трансакција. Сваки *chaincode* може да има дефинисану политику одобравања која одређује услове које је потребно испунити да би се трансакција сматрала одобреном.
3. Уређивач (енгл. *orderer*): више чворова уређивача формира уређивачку мрежу која је обезбјеђује канал за комуникацију између клијената и *peer* чворова и

нуди услуге прослијеђивања трансакција. Клијенти који се прикључе на канал могу да објаве трансакције које ће бити достављене свим *peer* чворовима и то у истом логичком редослиједу. Објављене трансакције представљају кандидате за додавање у дневник трансакција. Чворови уређивачи имају двије основне методе: *broadcast* и *deliver*.

- *broadcast*: клијент позива *broadcast* методу када жели да објави трансакцију. Као аргумент прослијеђује трансакцију коју жели да објави.
- *deliver*: уређивач користи *deliver* методу када жели да достави блок *peer* чворовима.

Извршавање трансакција почиње са креирањем трансакције од стране клијента и слањем приједлога трансакције, односно такозване *propose* поруке, одређеном скупу чворова који врше одобравање трансакција. Скуп чворова који врше одобравање је клијентима познат захваљујући политици одобравања која је дефинисан за *chaincode* који се позива. Чак иако се деси да је неки од тих чворова недоступан у моменту слања поруке, клијент који иницира трансакцију покушаће да задовољи политику одобравања уз помоћ доступних чворова.

Након пријема *propose* поруке, чвор који је задужен за одобравање прво провјерава валидност потписа клијента који је посало поруку а затим врши симулацију трансакције тако што је локално извршава. Извршавање трансакције подразумијева позивање *chaincode*-а који је референциран од стране трансакције. Као резултат симулације креира се скуп прочитаних (енгл. *readset*) и измијењених ресурса (енгл. *writeset*).

У колико је потпис клијента валидан и симулација трансакције прође успјешно чвор задужен за одобравање би требао да одлучи да одобри трансакцију. За одобрене трансакције чвор шаље потврду да је трансакција одобрена, односно *transaction endorsed* поруку, заједно са својим потписом клијенту који је иницирао трансакцију. У супротном, ако одбије да одобри трансакцију, чвор шаље *transaction invalid* поруку.

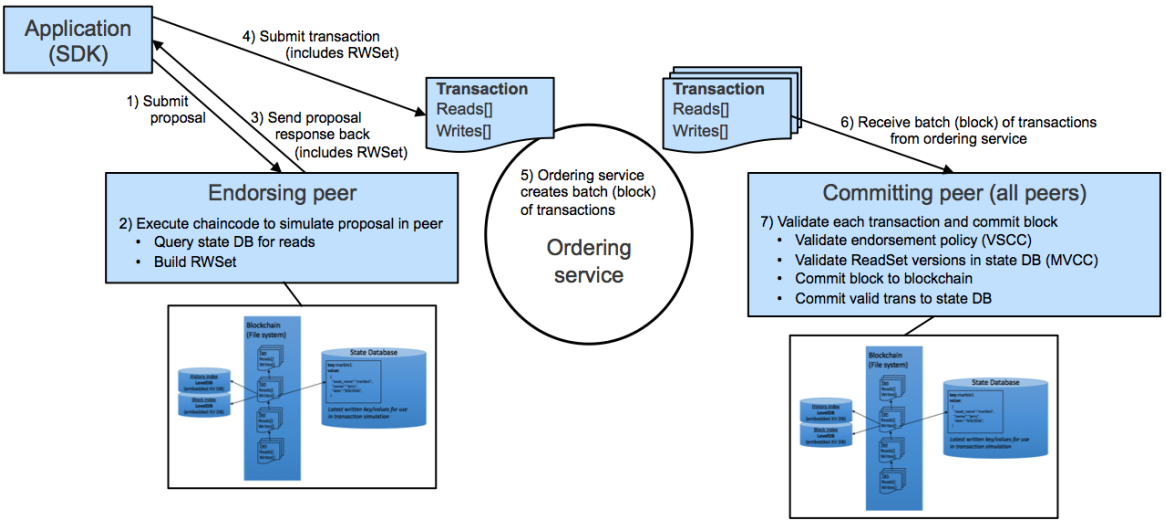
Клијент који је иницирао трансакцију чека да добије назад довољан број потврда да је трансакција одобрена. Довољан број потврда зависи од политике одобравања референцираног *chaincode*-а. У колико је политика задовољена трансакција се сматра одобреном али још увијек није додата у дневник трансакција. Ако не успије да сакупи потврде и задовољи политику одобравања, клијент одустаје од трансакције. Трансакције са довољним бројем потврда шаљу се уређивачима путем *broadcast* методе. Уређивач затим преузима трансакцију, уређује је унутар блока и доставља блок свим *peer* чворовима на мрежи путем *deliver* методе. Када се деси позив *deliver* методе и у колико је *peer* чвор обрадио све претходне *deliver* догађаје врше се следећи кораци:

- *Peer* провјерава да ли је задовољена политика одобравања за примљену трансакцију.
- Врши се провјера скупова прочитаних и измјењених ресурса да би се осигурало да ни један од тих ресура није измијењен у међувремену. Сваки ресурс у бази представљен је као кључ-вриједност пар и сваки од тих парова има ознаку тренутне верзије која се инкрементира сваки пут када се деси измјена датог

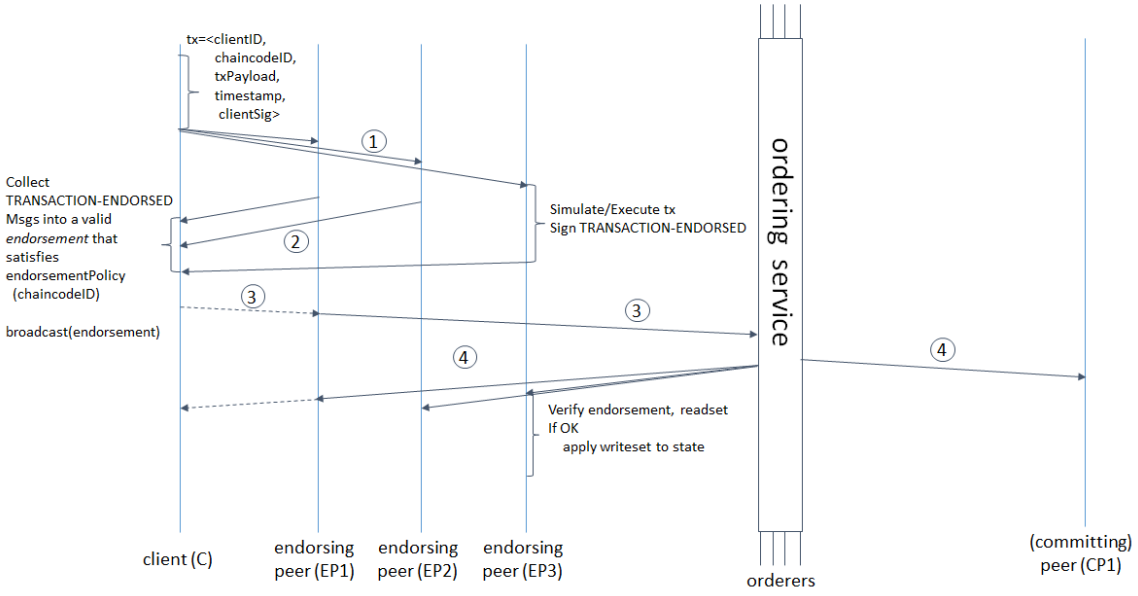
ресурса. Провјера скупова врши се тако што се провјеравају тренутне верзије ресурса са верзијама из скупва прочитаних и измијењених ресурса.

- У колико прођу све провјере, трансакција се означава као валидна, додаје се у дневник трансакције и вриједности из скупа измијењених ресура се примјењују на тренутно стање ресура. У супротном трансакција се означава као невалидна, ипак уписује у дневник трансакција али се настале измјене не примјењују на тренутно стање ресурса [8].

На сликама 12 [26] и 13 [8] приказан је ток одобравања трансакција у *Hyperledger Fabric* 1.0 верзији.



Слика 12. Ток одобравања трансакције у Hyperledger Fabric-у



Слика 13. Кораци приликом одобравања трансакције у Hyperledger Fabric-у

3.3. Поређење *Hyperledger Fabric*-а и *Ethereum*-а

Ethereum је још једна од имплементација *blockchain* технологије. Најзначајнија иновација коју је увео *Ethereum* јесте *Ethereum* виртуална машина (енгл. *Ethereum Virtual Machine EVM*) која представља окружење за извршавање паметних уговора на *blockchain* мрежи [27]. Захваљујући *Ethereum* виртуалној машини и паметним уговорима, *Ethereum* има јако широк спектар примјене. Умјесто развоја различитих *blockchain* мрежа, за различите примјене довољно је развити само различите апликације на једној заједничкој платформи. За разлику од *Hyperledger Fabric*-а, који има подршку за писање паметних уговора у *Go* и *Java* програмским језицима, *Ethereum* је развио сопствени Тјуринг комплетан језик за писање паметних уговора који се назива *Solidity.*

3.3.1. Циљни корисници

Двије основне разлике између *Ethereum*-а и *Hyperledger Fabric*-а јесу начин имплементације и циљни корисници. *Ethereum* је платформа која је више усмјерена ка дистрибуираним *B2C* (енгл. *Business to Consumer*) апликацијама код којих се очекује масовна употреба, односно апликацијама које су намјењене крајњим потрошачима и корисницима услуга. *Hyperledger Fabric,* са друге стране, има модуларну архитектуру која пружа велику флексибилност и усмјерена је ка *B2B* (енгл. *Business to Business*) апликацијама, односно апликацима које су намјењене за сарадњу пословних субјеката.

3.3.2. Консензус и повјерљивост

Ethereum виртуална машина и паметни уговори омогућавају развој великог броја различитих апликација са различитим намјенама али јавна природа *Ethereum* мреже и потпуна слобода приступа подацима онемогућавају задовољавање одређених захтјева који су неопходни у свијету пословних трансакција. Један од тих захтјева јесте скалабилност.

За постизање консензуса *Ethereum* користи доказ посла који је ресурсно веома захтјеван алгоритам. Због утицаја који рјешавање компјутерског задатка има на перформансе система *Ethereum* се окренуо ка истраживању других алгоритама. Тренутно је у развоју имплеметација алгоритма доказ посједовања коју су назвали Каспер [27].

Други захтјев јесте повјерљивост информација. Због употребе механизма за постизање консензуса који се заснива на доказу посла, код ког сви чланови мреже морају да се сложе око редослиједа и валидности трансакција, немогуће је подржати трансакције које би биле доступне само одређеним чланова мреже. Из тог разлога, за разлику од *Hyperledger Fabric*-а, *Ethereum* платформа није погодна за развој апликацијакоје захтијевају висок нико приватности и повјериљивости. *Hyperledger Fabric* је тај проблем ријешио увођењем канала који осигуравају да ће клијенти моћи видјети трансакције само оних канала на које су повезани.

3.3.3. Криптовалута

Још једна битна разлика између *Hyperledger Fabric*-а и *Ethereum*-а јесте постојање криптовалуте. *Hyperledger Fabric* нема криптовалуту уграђену у систем, док је за извршавање трансакција у *Ethereum*-у непходно је посједовање уграђене криптовалуте која се назива *Ether.* Свако извршавање паметног уговора у *Ethereum*-у троши одређену количину криптовалуте и та кооличина зависи од комплексности кода који се извршава. Чланови мреже криптовалуту зарађују приликом објаве новог блока као награду за учествовање у процесу консензуса. Постојање криптовалуте штити *Ethereum* од покушаја имплементирања паметних уговора који имају бесконачне петље. Извршавање таквог уговора ће се аутоматски зауставити након што се потроши сва количина криптовалуте додјељена том паметном уговору а све промјене проузроковане прије прекида ће бити поништене [28].

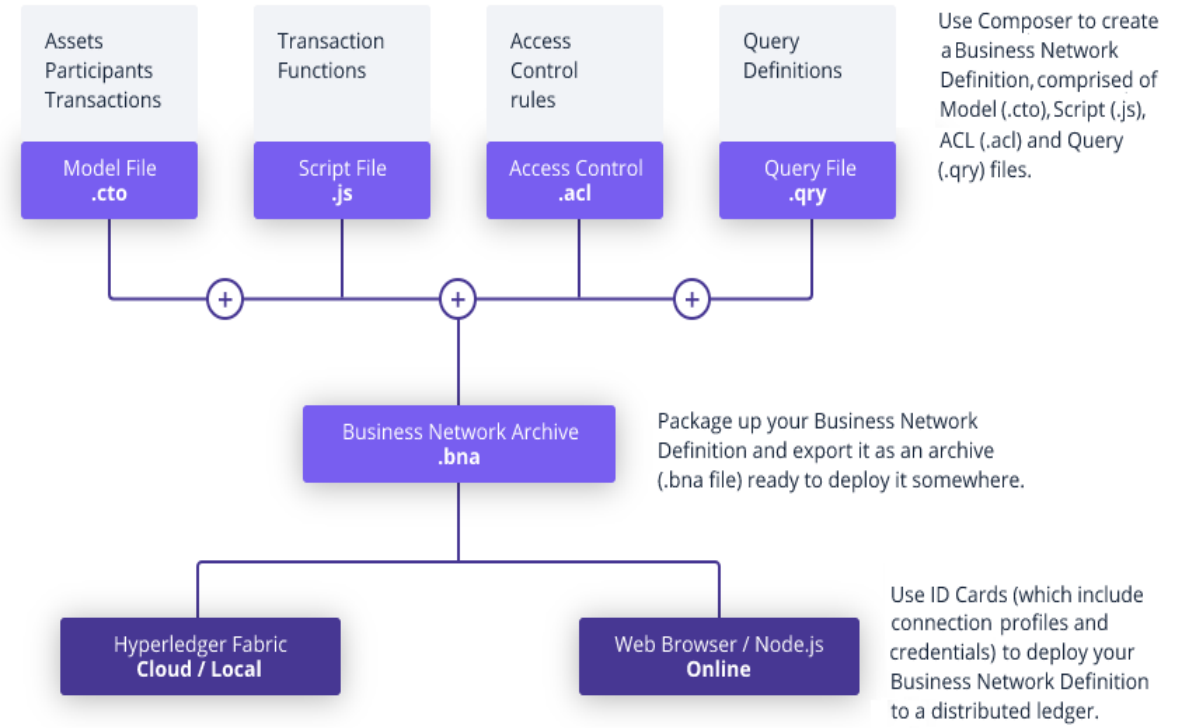
4. ОПИС СОФТВЕРСКОГ РЈЕШЕЊА

У овом поглављу опсисано је софтверско рјешење засновано на *Hyperledger Fabric blockchain* платформи, намјењено за примјену у индустрији видео игара. Такође је дат опис алата и технологија коришћених приликом развоја описаног софтверског рјешења.

4.1. Опис коришћених алата и технологија

За дефинисање спецификације *blockchain* мреже и генерисање клијентске апликације коришћен је *Hyperledger Composer* алат [29].

Hyperledger Composer је *open source* алат развијен са намјеном да олакша развој *blockchain* апликација за *Hyperledger Fabric* платформу. *Composer* омогућава корисницима да развијају апликације базиране на *Hyperledger Fabric*-у без познавања концепата спрецифичних за *blockchain* мреже. Примарни циљ алата јесте да се вријеме потребно за развој апликација смањи са неколико мјесеци на неколико недеља и да олакша интеграцију *blockchain* апликација са постојећим пословним системима [29]. Спецификација *blockchain* мреже, коришћењем *Hyperledger Composer*-а, подразумијева дефинисање модела који описује ресурсе унутар мреже, учеснике и операције које учесници могу извршавати над ресурсима, представљене у виду трансакција. Поред модела потребно је описати логику сваке од трансакција која је дата у моделу и дефинисати права приступа над ресурсима (енгл. *Access Control Lists*). Опис логике трансакција задаје се путем *JavaScript* програмског језика и заправо представља пословну логику система. На слици 14 приказане су компоненте мреже дефинисане помоћу *Composer*-а [29].



Слика 14. Компоненте мреже специфициране Hyperledger Composer алатом

Ресурси могу бити опипљива или неопипљива добра, услуге или имовина. Чувају се у засебним регистрима и морају имати јединствене идентификаторе. Поред идентификатора, који су обавезни, могу имати било какве и било колико атрибута. Атрибути могу бити представљени примитивним структурама података или у виду веза ка другим ресурсима, учесницима или трансакцијама из модела.

Учесници представљају чланове пословне мреже. Могу да посједују ресурсе и извршавају трансакције. Као и ресурси, чувају се у посебним регистрима и морају имати јединствене идентификаторе. Унутар *blockchain* мреже, учесници имају додјељене идентитете.

Трансакције представљају механизме помоћу којих учесници управљају ресурсима. Могу да имају атрибуте и моделују се заједно са ресурсима и учесницима. Све трансакције пропагирају се на ниво *Fabric*-а гдје се организују у блокове и чувају у дневнику трансакција.

Догађају се такође дефинишу током спецификације мреже на исти начин као и ресурси и учесници. Догађаји могу бити емитовани од стране трансакција да обавијесте корисничке апликације да је нешто одрађено са подацима у бази.

Упити се користе када се желе претраживати подаци из базе која чува стања ресурса унутар *blockchain* мреже. Упити се дефинишу током спецификације мреже и могу користити промјењиве и параметре за дефинисање комплексних услова претраге.

Употребом *Coposer*-а, унутар мреже могу се дифинисати и права приступа. Права приступа нуде контролу приступа над ресурима, дефинишу операције које учесници

могу вршити и над којим ресурсима. Такође описују и које трансакције, под којим условима, учесници имају право да извршавају. Језик за дефинисање права приступа је довољно богат и развијен да може искразати услове као што су, на примјер, да само власник возила може пренијети власништво над тим возилом неком другом учеснику.

Hyperledger Composer такође омогућава програмерима да брзо и једноставно креирају комплетна и заокружена *blockchain* софтверска рјешења [29]. То подразумијева пословну логику, описану у паметним уговорима, које је смјештена на *blockchain*-у, *REST* апликативне програмсе интерфејсе, који омогућавају корисничким апликацијам приступ пословној логици и саме корисничке апликације. *Hyperledger Composer REST* сервер аутоматски, на основу модела дефинисаног помоћу *Composer*-а, генерише *REST* методе које врше операције креирања, добављања, измјене и брисања ресурса и учесника, као и извршавања трансакција.

Composer такође нуди и интерактивни веб интерфејс који се назива *Playground* и служи за тестирање дефиниција пословних *blockchain* мрежа. *Online* верзија покреће и одржава мрежу унутар веб претраживача [29]. Поред *Playground*-а *Composer* нуди и интерфејс командне линије, који нуди велики избор команди за интеракцију са постојећим пословним мрежама.

4.2. Опис *blockchain* мреже

На слици 15 и слици 16 приказан је специфицирани модел *blockchain* мреже која служи за управљање ресурсима, односно куповину и размјену добара (енгл. *items*), у видео играма, на основу кога је имплементирано софтверско рјешење.

```
/**
 * Game assets management network definition.
 */
namespace com.myapp

enum ItemListingState {
  o IN_AUCTION
  o OWNED
}

participant Player identified by email{
  o String email
  o String firstName
  o String lastName
  o Double balance
}

asset Game identified by gameId{
  o String gameId
  o String name
  o String publisher
}

asset ItemType identified by typeId {
  o String typeId
  o String name
  o Double price
  --> Game game
}

asset Item identified by itemId {
  o String itemId
  o ItemListingState status
  --> ItemType itemType
  --> Player owner
}

asset ItemListing identified by listingId {
  o String listingId
  o Double minPrice
  o ItemListingState status
  --> Item item
  o Offer[] offers
}
```

Слика 15. Ресурси и учесници моделовани унутар blockchain мреже за управљање ресурсима у видео играма

```
transaction BuyTokens{
  --> Player buyer
  o Double quantity
}

transaction BuyItem {
  --> ItemType itemType
  --> Player buyer
}

transaction TradeItems{
  --> Player trader1
  --> Player trader2
  --> Item[] trader1items
  --> Item[] trader2items
  o Double trader1tokens
  o Double trader2tokens
}

transaction ListItem {
  -->Item item
  o Double minPrice
}

transaction Offer {
  o Double bidPrice
  --> ItemListing listing
  --> Player player
}

transaction CloseBidding {
  --> ItemListing listing
}
```

Слика 16. Трансакције моделоване унутар blockchain мреже за управљање ресурсима у видео играма

Дефинисани модел садржи један тип учесника, четири врсте ресурса и шест различитих трансакција. Играчи су моделовани као тип учесника *Player* и они представљају једине учеснике у мрежи. Сваки играч идентификован је преко *email* адресе, има име, презиме и тренутно стање рачуна, односно количину токена коју

посједује. Токени представљају средство плаћања и могу се користити приликом куповине разних добара или приликом размјене са другим играчима.

Апликација подржава размјену добара између различитих видео игара те је из тог разлога у моделу је дефинисан ресурс *Game*. Свака игра има скуп предефинисаних добара које играчи могу да купе. Та предефинисана добра моделована су путем *ItemType* ресурса. Поред *ItemType* дефинисан је и ресурс *Item* који представља конкретну инстанцу одређеног *ItemType*-а. Другим ријечима, *ItemType* представља апстрактан опис одређеног *Item*-а у ком су описани сви његови атрибути, као што су име, цијена и видео игра којој припада то добро. Свака инстанца ресурса *Item* посједује атрибут који указује на који *ItemType* се односи и свог власника који представља конкретног играча. Сваки играч има право читања над свим добрима, али само власници добара имају право измјене и брисања добара која посједују.

Играчи могу да врше куповину нових добара, директну размјену са другим играчима и да стављају добра које посједују на аукцију. Куповина нових добара врши се искључиво у замјену за токене и представљена је *BuyItem* трансакцијама. Трансакција има два атрибута, играча типа *Player* који врши куповину и класу добра, типа *ItemType*, које играч жели да купи.

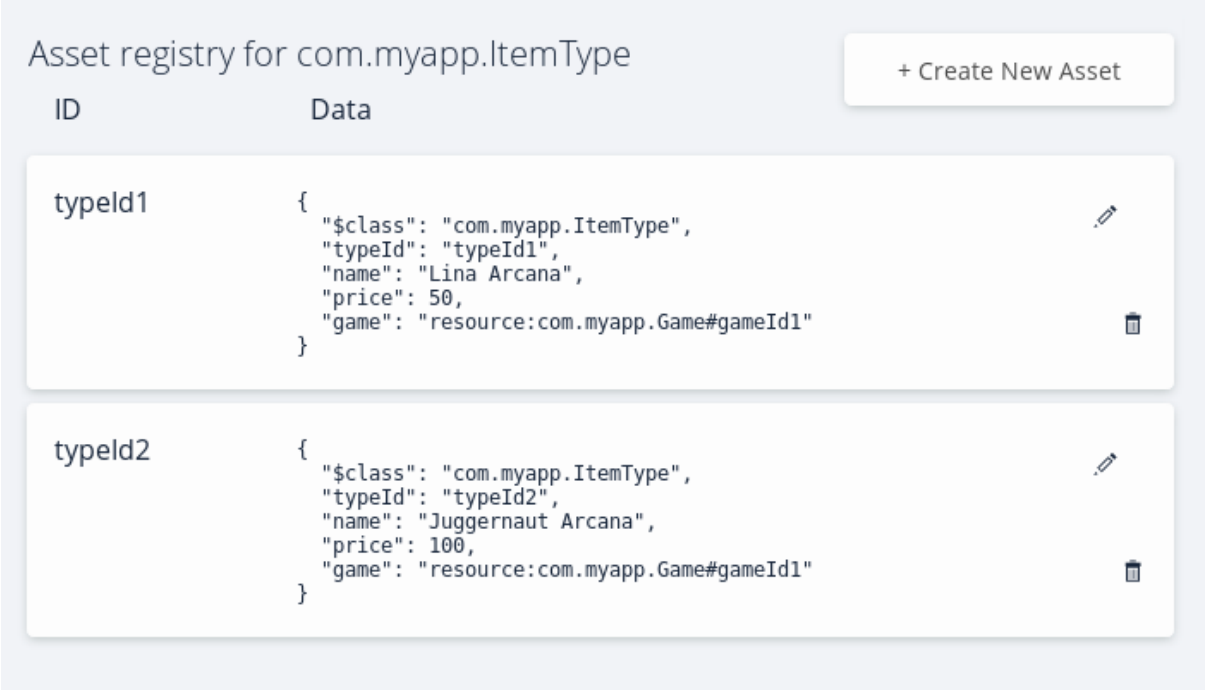
Играчи токене могу набавити на два начина. Куповина токена новцем представљена је *BuyTokens* трансакцијом која као атрибуте има играча који купује токене и количину токена коју жели да купи. Други начин јесте продаја добара другим играчима. То се такође може постићи на два начина. Први начин је путем *TradeItems* трансакција. *TradeItems* трансакција представља директну размјену између два играча. Трансакција као атрибурте има играче који учествују у размјени и за сваког играча листу добара и количину токена које нуди у замјену за листу добара и количину токена другог играча.

Други начин на који играчи могу набавити токене јесте постављање добара на аукцију. То се врши путем *ListItem* трансакција, које креирају нове инстаце *ItemListing* ресурса. *ItemListing* као атрибуте садржи јединствени идентификатор, минималну цијену за коју добро може да се прода, тренутни статус аукције, добро који се продаје и низ понуда за дату аукцију. Куповина добара која се налазе на аукцији врши се искључиво путем понуда, које су представљене *Offer* трансакцијама. *Offer* трансакција има следеће атрибуте: играча, типа *Player*, који даје понуду, аукцију за оно добро које тај играч жели да купи, типа *ItemListing*, и количину токена која представља понуду. Затварање аукције и продаја добара врше се позивом *CloseBidding* трансакције.

Тестирање мреже представњене у овом раду извршено је унутар интерактивног веб интерфејса, *Playground*-а, који представља једну од компоненти *Hyperledger Composer*-а. Са циљем тестирања дефинисаних трансакција и права приступа, унапријед је ручно креирано неколико учесника класе *Player* и неколико ресурса класа *Game* и *ItemType.* На сликама 17, 18 и 19 приказани су унапријед креирани ресурси и учесници.

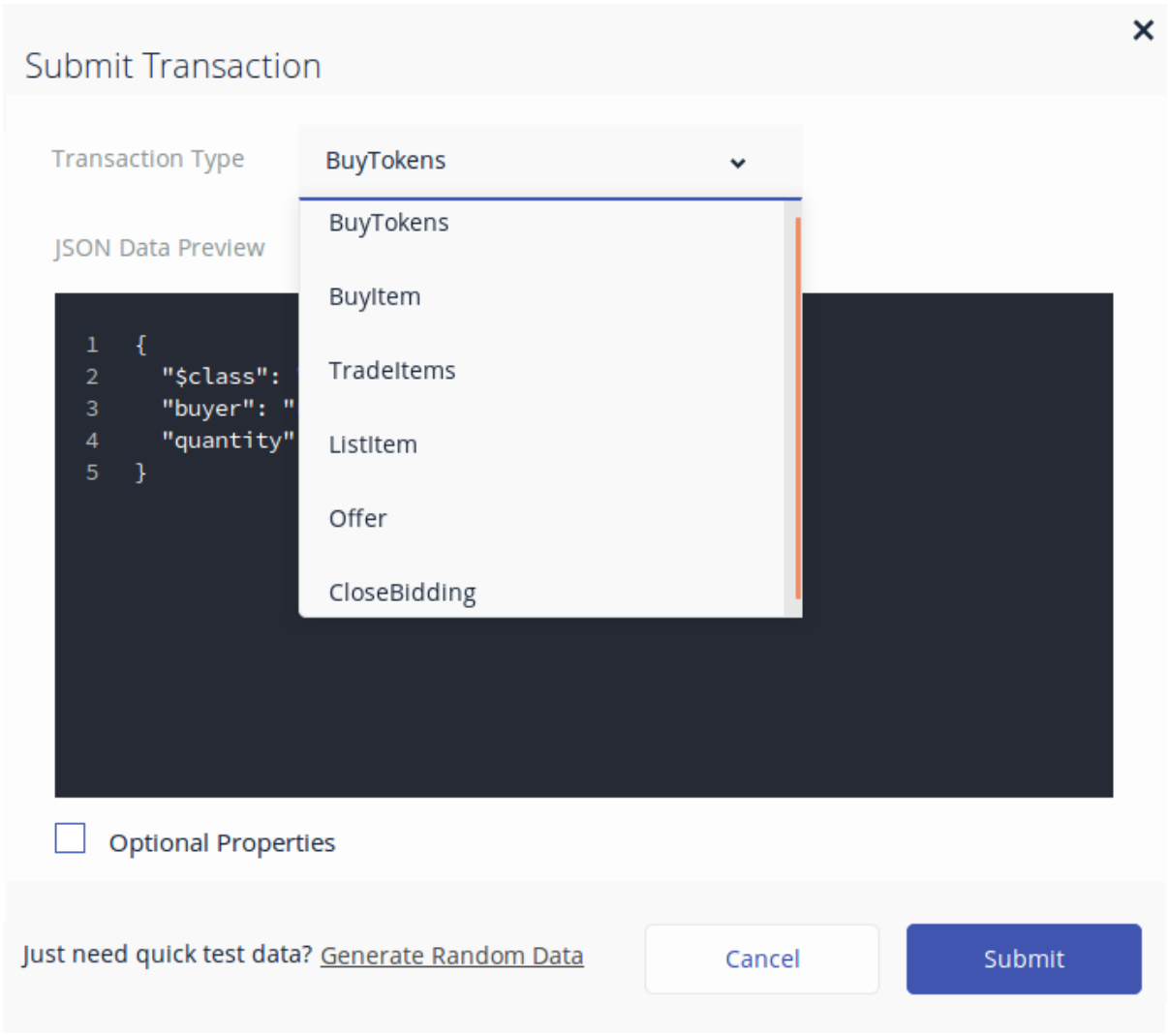
Слика 17. Унапријед креирани учесници у мрежи

Слика 18. Унапријед креирани Game ресурси



Слика 19. Унапријед креирани ItemType ресурси

Као што је већ поменуто, у моделу је специфицирано шест различитих трансакција које могу да се извшавају на мрежи. На слици 20 приказана је форма унутар *Playground*-а помоћу које корисник може да иницира дефинисане трансакције.

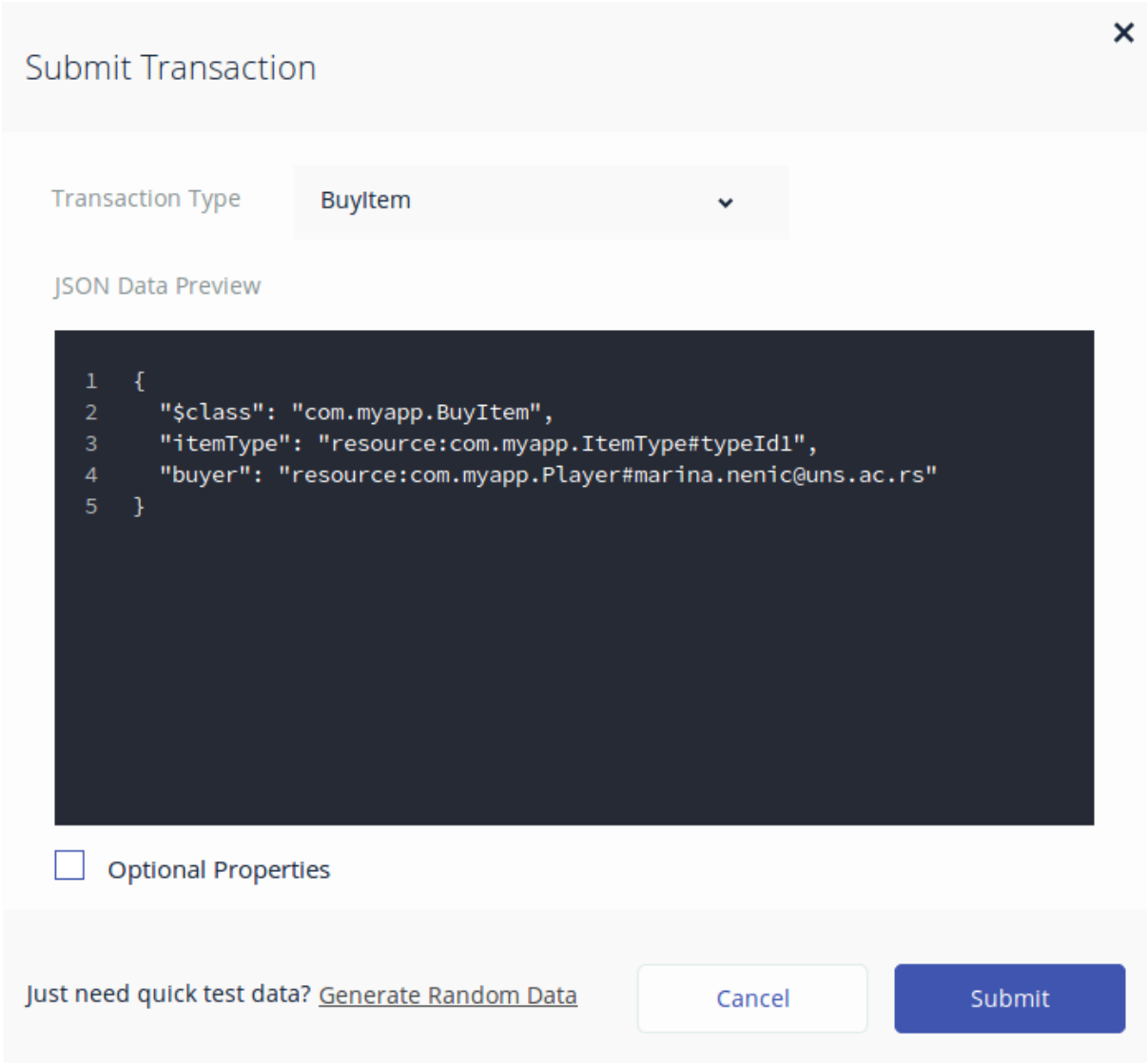


Слика 20. Форма за иницирање трансакција и листа свих понуђених трансакција које се могу извршавати

Играчи могу да врше куповину токена, који касније могу бити искоришћени приликом куповине добара, давања понуда на аукцијама или размјене. Куповина токена врши се преко *BuyTokens* тансакација која има два атрибута: играча који врши куповину и количину токена коју жели да купи. Када се трансакција изврши, стање токена на рачуну играча повећава се за ону количину која је дата као атрибут трансакције.

BuyItem трансакцијама представљена је куповина нових добара. Трансакција има два атрибута, играча типа *Player* који врши куповину и класу добра, типа *ItemType*, које играч жели да купи. Приликом извршавања трансакције систем провјерава стање рачуна играча који је дат ако атрибут трансакције и уколико играч посједује довољну количину токена креира се нова инстанца класе *Item*. Играч се поставља као власник креираног добра и стање токена на његовом рачуну се умањује за ону количину која је дефинисана цијеном класе добра, која је дата као други атрибут трансакције. Статус добра се приликом креирања подешава на „*OWNED*“ и служи да осигура да се једно добро не може поставити на више аукција истовремено. На сликама 21, 22 и 23

приказана је форма за покретање *BuyItem* трансакције унутар *Playground*-а, ново добро креирано током извршавања и ново стање токена на рачуну играча.



Слика 21. Иницирање BuyItem трансакције

Слика 22. Нова инстанца ресурса Item креирана извршавањем BuyItem трансакције

Слика 23. Ново стање токена на рачуну играча

TradeItems трансакција представља размјену између два играча и има шест атрибута: играче који учествују у размјени и за сваког играча листу добара и количину токена које нуди у замјену за листу добара и количину токена другог играча. Сви атрибути, осим играча, су опциони. То значи да *TradeItems* трансакција подржава било који од следећих случајева: замјена једног или више добара једног играча за једно или више добара другог играча, замјена једног или више добара једног играча за одређену количину токена другог играча, замјена једног или више добара једног играча за једно или више добара и одређену количину токена другог играча, као и опцију поклањања једног или више добара или токена једног играча другом играчу. Приликом извршавања трансакције, систем провјерава оба атрибута која се односе на токене. У колико је задати број токена негативан или играч који даје токене приликом размјене нема довољну количину токена на рачуну систем ће пријавити грешку. Ако провјера токена прође биће извршена размјена. Стање токена на рачуну сваког од играча биће умањено за ону количину коју је он понудио, а повећано за ону количину коју је други играч понудио у размјени, а власништва над добрима из листи добара, које су дате као атрибути, ће бити замијењена.

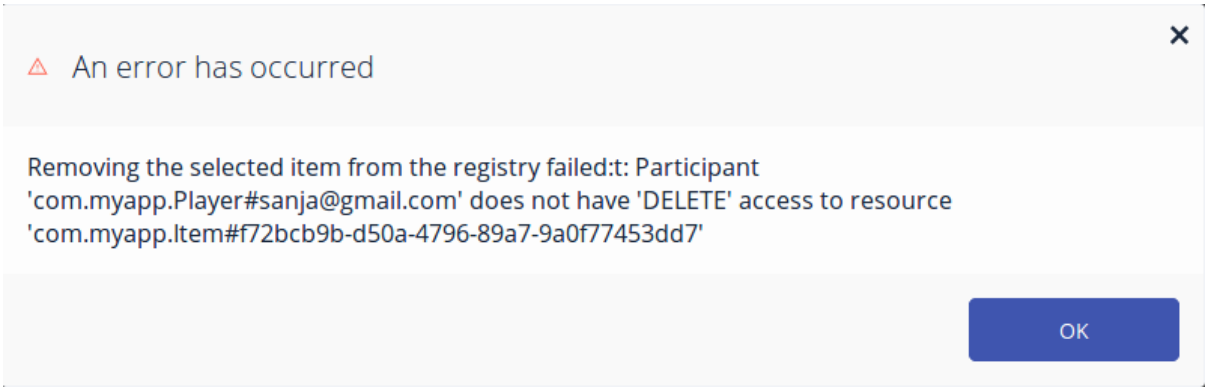
ListItem трансакције омогућавају постављање добара на аукције. Трансакције имају два атрибута: добро које се ставља на аукцију и минималну цијену по којој се дато добро може продати. Приликом извршавања *ListItem* трансакције провјерава се статус добра које се поставља на аукцију. Ако је статус добра "*IN-AUCTION"* то значи да се добро већ налази на аукцији и биће пријављена грешка. У супротном, креира се нова инстанца *ItemListing* ресурса. *ListItem* трансакције могу извршавати само они играчи који су власници добара која се стављају на аукције.

Куповина добара са аукције врши се коришћењем *Offer* и *CloseBidding* трансакција. *Offer* трансакције представљају давање понуда и има три атрибута: играча који даје понуду, аукцију са које жели да купи добро и количину токена која представља понуду. Систем упоређује количину токена на рачуну играча са задатом понудом и уколико играч не посједује довољну количину биће пријављена грешка. У супротном се провјерава да ли је понуда већа од минималне цијене по којој се добро на аукцији може купити. Ако ни приликом не провјере не буде пријављена грешка, понуда, у виду *Offer* трансакције, ће бити додата у листу понуда аукције која је задата као атрибут трансакције.

Аукција ће бити активна и играчи могу давати понуде за ту аукцију све док се не позове *CloseBidding* трансакција за дату аукцију. Право извршавања *CloseBidding* трансакције, као код *ListItem* трансакце, има само играч који је власник добра које је на аукцији. Приликом *CloseBidding* трансакције систем узима највећу понуду из листе

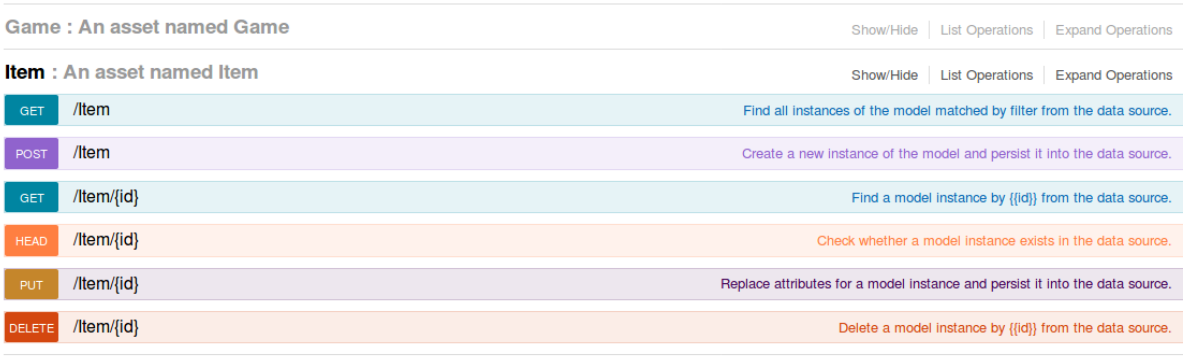
понуда, провјерава стање токена на рачуну играча који је задао дату понуду и уколико играч посједује довољну количину токена додјељује му добро који се налази на аукцији. Датом играчу се стање токена на рачуну умањује а претходном власнику, односно играчу који је поставио добро на аукцију, додјељују се токени који представљају највећу понуду. Статус аукције и добра које се налазило на аукцији, приликом извршавања *CloseBidding* трансакције, поставља се на „*OWNED*“. У колико се деси да је иницирана *CloseBidding* трансакција а аукција нема ни једну понуду, или ни једна понуда не може да буде прихваћена због недовољне количине токена на рачунима играча, аукција ће бити затворена и стање аукције и добра ће бити промјењено али се неће десити промјена власника добра, као ни стања токена на рачуну играча који је поставио добро на аукцију.

Поред извршавања трансакција, у *Playground*–у су тестирана и права приступа ресурсима и права извршавања трансакција. У колико играч покуша да измијени, обрише или стави на аукцију неко добро које му не припада, систем ће пријавити грешку. Грешка ће такође бити пријављена и у случају покушаја покретања *CloseBidding* трансакције од стране играча који није власник добра које је на аукцији. На слици 24 приказана је грешка која је пријављена приликом покушаја извршавања операција за које корисник који их покреће нема правво извршавања.

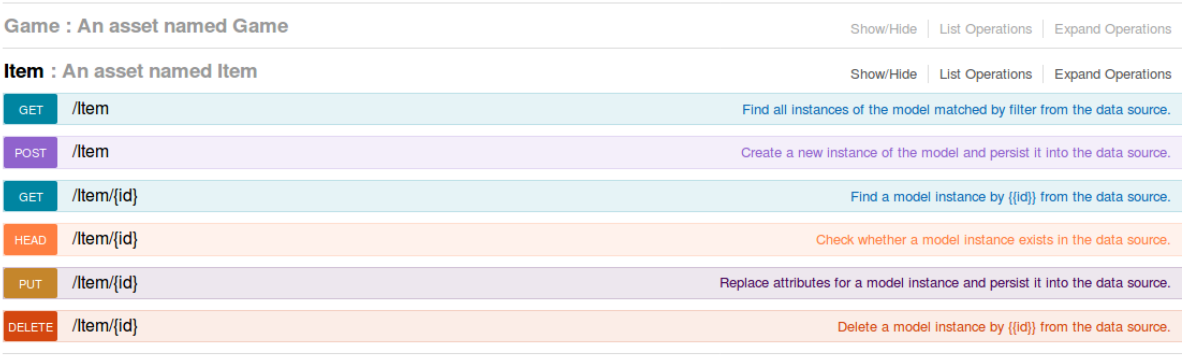


Слика 24. Пријава грешке приликом покушаја извршавања операције за који учесник нема право извршавања

На основу спецификације мреже, *Hyperledger Composer* мапира описане трансакције на генеричке паметне уговоре који се налазе на *Hyperledger Fabric blockchain* мрежи и генерише *REST* сервисе помоћу којих се позивају дати паметни уговори. За сваки ресур и учесника мреже, генерише се шест *REST* метода које служе за добављање свих ресурса или учесника на мрежи, за креирање нових инстанци ресурса или учесника, као и добављање, измјену, брисање и провјеру постојања појединачних ресурса или учесника. На сликама 25 и 26 приказане су генерисане *REST* методе за ресурсе и учеснике.

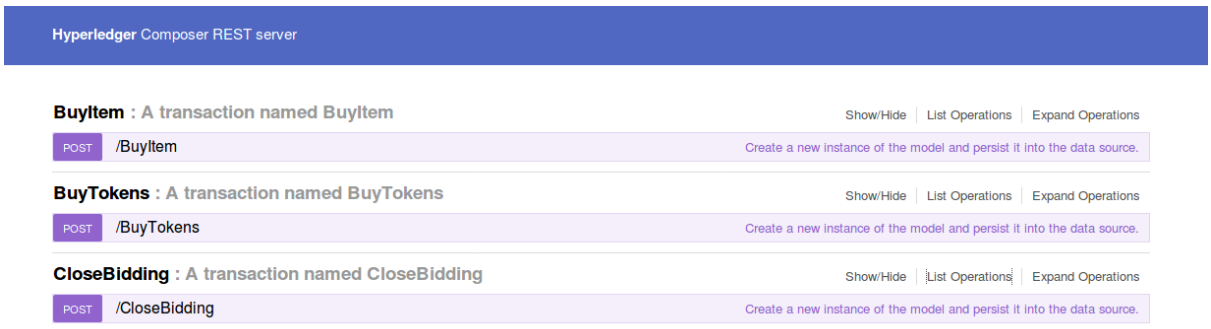


Слика 25. Генерисане REST методе за ресурсе

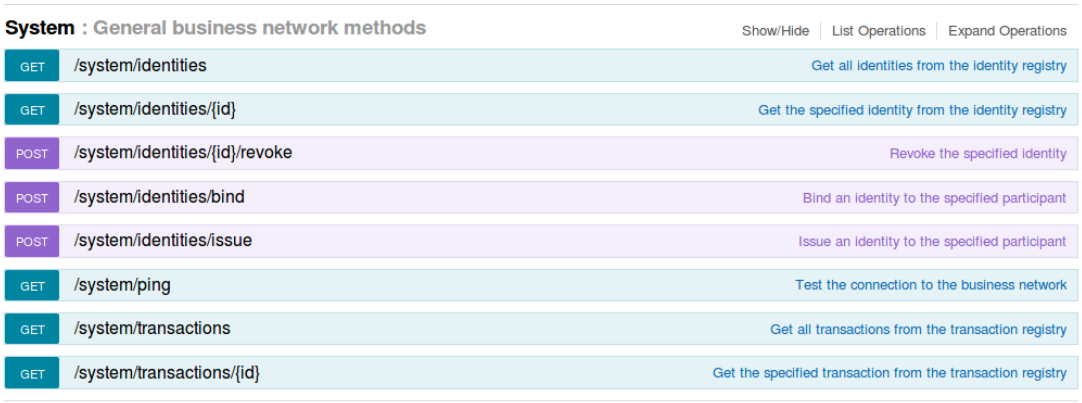


Слика 26. Генерисане REST методе за учеснике

У случају трансакција, генерише се само један метод који служи за извршавање трансакције, односно креирање нове инстанце трансакције. Могуће је вршити и добављање свих трансакција, или појединачних путем јединственог идентификатора, али су те методе приликом генерисања сврстане међу системске *REST* методе. Поред метода за добављање трансакција, у оквиру системских метода генеришу се и методе за добављање идентитета, креирање нових и додјелу постојећих идентитета учесницима, укидање идентитета и провјеру конекције ка мрежи. На сликама 27 и 28 приказане су *REST* методе за извршавање трансакција и системске *REST* методе.

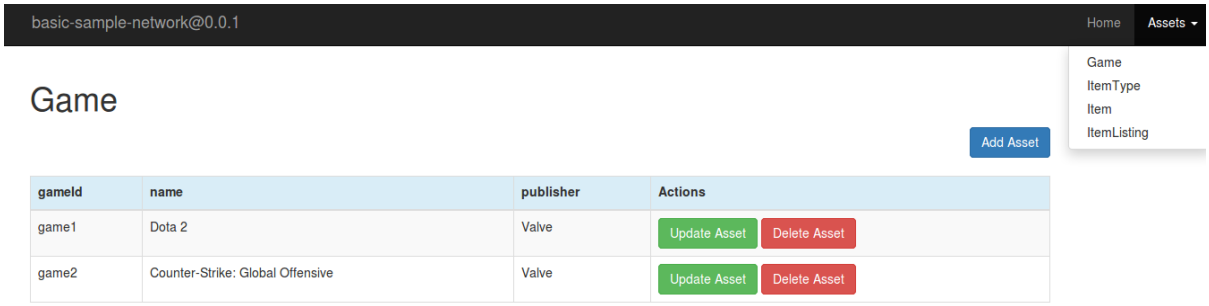


Слика 27. Генерисане REST методе за трансакције

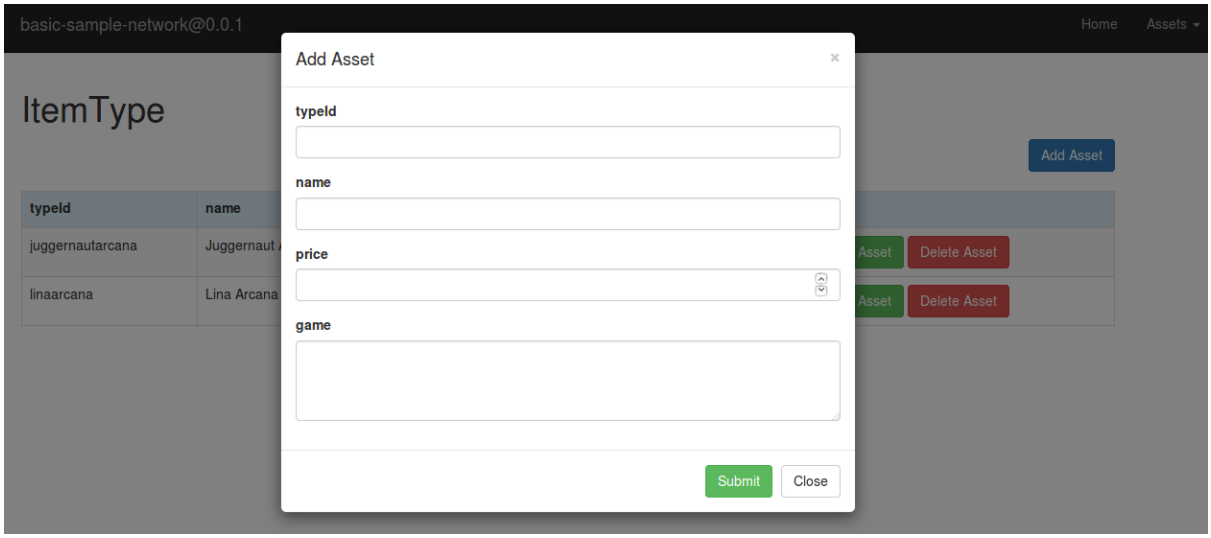


Слика 28. Генерисане системске REST методе

Генерисани *REST* сервиси могу се користити и позивати из корисничких апликација. *Hyperledger Composer* нуди опцију генерисања костура *Angular 2* веб апликације која позива функционалности паметних уговора на *blockchain* мрежи путем *REST* сервиса [29]. Апликација нуди операције читања, креирања, измјене и брисања ресурса дефинисаних моделом и представља одличану основу за развој софтверских рјешења. На слици 29 дат је приказ излистаних *Game* ресурса у генерисаној веб апликацији а слика 30 приказује форму за додавање новог ресурса класе *ItemType.*



Слика 29 Приказ Game ресурса



Слика 30. Форма за додавања новог ItemType ресурса

5. ЗАКЉУЧАК

У овом раду описане су основе *blockchain* дистрибуираних система и предложено је једно софтверско рјешење за њихову примјену у индустрији видео игара. Представљена је спецификација *blockchain* мреже која служи за управљање ресурсима у видео играма. Унутар мреже моделован је један тип корисника, а то су играчи и представљени су *Player* класом. Учесници унутар мреже могу да рукују ресурсима путем трансакција. Као ресурси унутар мреже описане у овом раду моделоване су видео игре, представљене класом *Game*, класе добара у видео играма, представљене класом *ItemType,* конкретна добра додјељена играчима, представљена класом *Item,* и добра на аукцијама, представљена класом *ItemListing.* Основни циљ мреже јесте да омогући куповање и размјену добара између играча у видео играма. Из тог разлога моделовано је шест трансакција: *BuyTokens, BuyItem, TradeItems, ListItem, Offer* и *CloseBidding. BuyTokens* трансакција представља куповину токена који представљају замјену за новац у мрежи. *BuyItem* трансакција представља куповину добара у замјену за токене. Приликом *BuyItem* трансакција креирају се нове инстанце добара док се у размјени и у аукцијама могу користити само постојећа добра. *TradeItems* трансакција представља размјену добара између два играча. Трансакција је моделована тако да подржава различите случајеве током размјене: замјена једног добра за друго, замјена више добара за једно или више добара, као и токена. Помоћи *TradeItems* трансакција играчи такође могу поклањати добра другим играчима. Поред директне размјене између два играча, играчи такође могу размјењивати добра путем аукција. Играч може да постави добро које посједује на аукцију и то се обавља помоћу *ListItem* трансакција. Путем *Offer* трансакција, играчи шаљу понуде за одређено добро које се налази на аукцији. Када се аукција затвори, узима се највећа понуда и играч који је дао дату понуду купује добро које се налазило на аукцији. Затварање аукција врши се путем *CloseBidding* трансакција.

За спецификацију модела коришћен је *Hyperledger Composer* а за имплементацију *blockchain*-а на којој ће бити подигнута мрежа изабран је *Hyperledger Fabric*. *Fabric* се фокусирао на задовољавање великог броја индустријских захтјева и нуди јако широк спектар примјене али то са собом носи одређене недостатке као што је тежина употребе. Поред тога што је и сам *blockchain* релативно нова и неистражена технологија, употребу технологије додатно отежава и мноштво опција које *Hyperledger* нуди. Процес подешавања и покретања мреже је јако мукотрпан и слабо документован, те представља велику препреку у развоју апликација. Ту се *Composer* показао као јако користан алат јер процес развоја апликација и дефинисања мреже диже на већи ниво апстракције. Умјесто да се дефинишу концепти на нивоу *blockchain*-а, као што су чворови мреже и паметни уговори, *Composer* пружа могућност дефинисања мреже на нивоу учесника и трансакција што је и корисницима и програмерима доста познатије и природније. Међутим ни *Composer* није идеално рјешење јер представља алат који је још увијек у процесу развоја и не покрива у потпуности све функционалности које *Hyperledger Fabric* нуди.

Из тог разлога, планови за даљи рад рад укључују развој одређених функционалности софтверског рјешења директно на *Hyperledger Fabric*-у, као што су

дефинисање паметних уговора и политика одобравања, као и повезивање имплементираног софтверског рјешења са апликативним програмским интерфејсима (енгл. *Application Programming Interface - API*) конкретних видео игара.

ЛИТЕРАТУРА

[1] *Hyperledger Fabric Documentation*, доступно на: http://hyperledger-fabric.readthedocs.io/en/latest/ , последњи приступ 26.09.2017.

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

[3] Narayanan A., Bonneau J., Felten E., Miller A., Goldfeder S., *Bitcoin and Cryptocurrency Technologies*, Princeton Unversity Press, 2016.

[4] Gupta M., *Blockchain for Dummies*, *IBM Limited Edition*, John Wiley & Sons, 2017.

[5] *Peer-to-peer*, Wikipedia, the free encyclopedia, https://en.wikipedia.org/wiki/Peer-to-peer, последњи приступ 16.09.2017.

[6] *Smarter banking*, https://cmp.smu.edu.sg/ami/article/20161208/smarter-banking, последњи приступ 16.09.2017.

[7] *Blockchain in financial services*, https://www.pwc.com/us/en/financial-services/research-institute/blockchain.html , последњи приступ 16.09.2017.

[8] *Next Consensus Architecture Proposal* https://github.com/hyperledger/fabric/blob/master/proposals/r1/Next-Consensus-Architecture-Proposal.md#32-transaction-evaluation-against-endorsement-policy , последњи приступ 16.09.2017.

[9] *Why we should drop the whole "Bitcoin vs blockchain" discussion*, https://medium.com/@YacineGhalim/why-we-should-drop-the-whole-bitcoin-vs-blockchain-discussion-e3e38e9a5104, последњи приступ 16.09.2017.

[10] *Understanding Blockchain Technology*, https://www.linkedin.com/pulse/understanding-blockchain-technology-nassir-khan последњи приступ 16.09.2017.

[11] *Merke tree*, Wikipedia, the free encyclopedia, https://en.wikipedia.org/wiki/Merkle_tree, последњи приступ 16.09.2017.

[12] *Sybil attack*, Wikipedia, the free encyclopedia, https://en.wikipedia.org/wiki/Sybil_attack, последњи приступ 16.09.2017.

[13] Baliga A., *Understanding Blockhain Consensus Model*, 2017 https://www.persistent.com/wp-content/uploads/2017/04/WP-Understanding-Blockchain-Consensus-Models.pdf, последњи приступ 16.09.2017.

[14] *Consensus Mechanisms used in Blockchain*, https://www.linkedin.com/pulse/consensus-mechanisms-used-blockchain-ronald-chan последњи приступ 16.09.2017.

[15] *Review of blockchain consensus mechanisms*, https://blog.wavesplatform.com/review-of-blockchain-consensus-mechanisms-f575afae38f2 , последњи приступ 16.09.2017.

[16] *51% Attack*, https://learncryptography.com/cryptocurrency/51-attack, последњи приступ 16.09.2017.

[17] *What is a Smart Contract*, https://bol.bna.com/what-is-a-smart-contract/, последњи приступ 16.09.2017.

[18] *Smart contract*, Wikipedia, the free encyclopedia, https://en.wikipedia.org/wiki/Smart_contract, последњи приступ 16.09.2017.

[19] *Smart contracts*, https://blockchainhub.net/smart-contracts/, последњи приступ 16.09.2017.

[20] *Hyperledger - Blockchain Technologies for Business*, https://www.hyperledger.org/ , последњи приступ 16.09.2017.

[21] *Hyperledger Architecture, Volume 1*, https://www.hyperledger.org/wp-content/uploads/2017/08/HyperLedger_Arch_WG_Paper_1_Consensus.pdf , последњи приступ 16.09.2017.

[22] *What is Hyperledger Fabric*, http://hyperledger-fabric.readthedocs.io/en/latest/blockchain.html последњи приступ 16.09.2017.

[23] *Hyperledger Whitepaper*, http://www.the-blockchain.com/docs/Hyperledger%20Whitepaper.pdf последњи приступ 16.09.2017.

[24] *Membership Services*, https://github.com/christo4ferris/node_sdk/blob/master/membership-services.md последњи приступ 16.09.2017.

[25] *Learn Chaincode*, https://github.com/IBM-Blockchain-Archive/learn-chaincode, последњи приступ 16.09.2017.

[26] *Top 6 techical advantages of Hyperledger Fabric for blockchain networks*, https://www.ibm.com/developerworks/cloud/library/cl-top-technical-advantages-of-hyperledger-fabric-for-blockchain-networks/index.html, последњи приступ 16.09.2017.

[27] *Proof of Stake FAQ*, https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ, последњи приступ 16.09.2017.

[28] *Ethereum Whitepaper, A Next-Generation Smart Contract and Decentralized Application Platform* https://github.com/ethereum/wiki/wiki/White-Paper, последњи приступ 16.09.2017.

[29] *Hyperledger Composer Documentation*, доступно на: https://hyperledger.github.io/composer/introduction/introduction.html, последњи приступ 26.09.2017.

ДОДАТАК А

У овом додатку приказан је *JavaScript* код који представља пословну логику *blockchain* мреже која је описана у овом раду. Листинг 1. приказује функцију која описује логику *BuyTokens* трансакције.

```
/**
 *
 * @param {com.myapp.BuyTokens} tx.
 * @transaction
 */
function BuyTokens(tx) {

  if (tx.quantity < 0) {
    throw new Error('Token quantity must be a positive number!');
  }

  // Update the player balance with the new value.
  tx.buyer.balance += tx.quantity;

  // Get the participant registry for the player.
  return getParticipantRegistry('com.myapp.Player')
    .then(function (playerRegistry) {
    // Update the asset in the asset registry.
    return playerRegistry.update(tx.buyer);
  });

}
```

Листинг 1. Логика *BuyTokens* трансакције

```
/**
 *
 * @param {com.myapp.BuyItem} tx.
 * @transaction
 */
function BuyItem(tx) {

  if (tx.buyer.balance < tx.item.price) {
    throw new Error('Not enough tokens for the purchase!');
  }

  // Update the player balance with the new value.
  tx.buyer.balance -= tx.itemType.price;

  // Create an item instance and assign it to the buyer
  var factory = getFactory();
  var item = factory.newResource('com.myapp', 'Item', tx.transactionId);
  item.type = tx.itemType;
  item.owner = tx.buyer;
  item.status = 'OWNED';

  // Get the participant registry for the player.
  return getParticipantRegistry('com.myapp.Player')
    .then(function (playerRegistry) {
    // Update the paler in the player registry.
    return playerRegistry.update(tx.buyer);
  })
  .then(function() {
    // Get the asset registry for the item.
    return getAssetRegistry('com.myapp.Item')
  })
  .then(function(itemRegistry){
    // Add the new item to the registry.
    return itemRegistry.add(item);
  });

}
```

Листинг 2. Логика *BuyItem* трансакције

```
/**
 *
 * @param {com.myapp.TradeItems} tx.
 * @transaction
 */
function TradeItems(tx) {

  // If the first trader is trading with tokens check his balance
  if (tx.trader1tokens < 0) {
    throw new Error('Token number must be positive!');
  }
  else if (tx.trader1tokens > tx.trader1.balance) {
    throw new Error('Not enough tokens to trade!');
  }

  // If the second trader is trading with tokens check his balance
  if (tx.trader2tokens < 0) {
    throw new Error('Token number must be positive!');
  }
  else if (tx.trader2tokens > tx.trader2.balance) {
    throw new Error('Not enough tokens to trade!');
  }

  // Update first and second trader's balances
  tx.trader1.balance -= tx.trader1tokens;
  tx.trader1.balance += tx.trader2tokens;

  tx.trader2.balance -= tx.trader2tokens;
  tx.trader2.balance += tx.trader1tokens;

  //Update the ownership of the traded items
  forEach(item in tx.trader2items)
    item.owner = tx.trader1;

  forEach(item in tx.trader1items)
    item.owner = tx.trader2;

  var items = []
  items.concat(trader1items);
  items.concat(trader2items);

  // Get the participant registry for the player.
  return getParticipantRegistry('com.myapp.Player')
    .then(function (playerRegistry) {
    // Update the player in the player registry.
    return playerRegistry.updateAll([tx.trader1, tx.trader2]);
  })
  .then(function() {
    // Get the asset registry for the item.
    return getAssetRegistry('com.myapp.Item')
  })
  .then(function(itemRegistry){
    // Update the items in the item registry.
    return itemRegistry.updateAll(items);
  });

}
```

Листинг 3. Логика *TradeItems* трансакције

```
/**
 *
 * @param {com.myapp.ListItem} tx.
 * @transaction
 */
function ListItem(tx) {

  if (tx.item.status == 'IN-AUCTION') {
    throw new Error('Item is already auctioned!');
  }

  var factory = getFactory();
  var listing = factory.newResource('com.myapp', 'ItemListing', tx.transactionId);
  listing.item = tx.item;
  listing.minPrice = tx.minPrice;
  listing.status = 'IN-AUCTION';
  listing.item.status = 'IN-AUCTION';

  return getAssetRegistry('com.myapp.ItemListing')
      .then(function(itemListing) {
          // save the listing
          return itemListing.add(listing);
      });
}
```

Листинг 4. Логика *ListItem* трансакције

```
/**
 *
 * @param {com.myapp.Offer} offer.
 * @transaction
 */
function Offer(offer) {

  var listing  =  offer.listing;
  if (listing.status !== 'IN-AUCTION') {
    throw new Error('Item is not auctiond!');
  }

  if (offer.player.balance < offer.bidPrice)
    throw new Error('Not enough tokens to bid!');

  if (offer.bidPrice < offer.listing.minPrice)
    throw new Error('Offer  too small!');

  if(listing.offers == null) {
    listing.offers = [];
  }

  listing.offers.push(offer);
  return getAssetRegistry('com.myapp.ItemListing')
      .then(function(itemListing) {
          // save the listing
          return itemListing.update(listing);
      });
}
```

Листинг 5. Логика *Offer* трансакције

```
/**
 *
 * @param {com.myapp.CloseBidding} closeBidding
 * @transaction
 */
function closeBidding(closeBidding) {

    var listing = closeBidding.listing;
    if (listing.status !== 'IN-AUCTION') {
        throw new Error('Item is not auctioned');
    }

    var highestoffer = null;
    var buyer = null;
    var seller = listing.item.owner;
    listing.status = 'OWNED';
    listing.item.status = 'OWNED';
    if (listing.offers && listing.offers.length > 0) {
        // sort the bids by bidPrice
        listing.offers.sort(function(a, b) {
            return (b.bidPrice - a.bidPrice);
        });

        for (var i = 0; i < listing.offers.length; i++){

          highestoffer = listing.offers[i];
          buyer = highestoffer.player;
          if (buyer.balance >= highestoffer.bidPrice) {
            seller.balance += highestoffer.bidPrice;
            buyer.balance -= highestoffer.bidPrice;
            listing.item.owner = buyer;
            listing.offers = null;
            break;
          }
        }
    }
    return getAssetRegistry('com.myapp.Item')
        .then(function(itemRegistry) {
            return itemRegistry.update(listing.item);
        })
        .then(function() {
            return getAssetRegistry('com.myapp.ItemListing')
        })
        .then(function(itemListingRegistry) {
            return itemListingRegistry.update(listing);
        })
        .then(function() {
            return getParticipantRegistry('com.myapp.Player')
        })
        .then(function(playerRegistry) {
            // save the buyer and seller
            return playerRegistry.updateAll([buyer, seller]);
        });
}
```

Листинг 6. Логика *CloseBidding* трансакције

СПИСАК КОРИШЋЕНИХ СКРАЋЕНИЦА

Скраћеница	Значење
SHA	Secure Hash Algorithm – сет криптографских хеш функција
EVM	Ethereum Virtual Machine – окружење за извршавање паметних уговора на blockchain мрежи развијено од стране Ethereum-a
MSP	Membership Service Provider – компонента Hyperledger Fabric-a која апстрахује криптографске механизме који се користе приликом издавања сетрификата
B2C	Business-to-Consumer – модел мрежног пословања код ког је продаја робе или прижање услуга усмјерено ка крајњим потрошачима, путем интернета
B2B	Business-to-Business – модел мрежног пословања који је усмјерен ка сарадњи између пословних субјеката
REST	Representational State Transfer – механизам за постизање интероперабилности између рачунарских система на интернету
API	Application Programming Interface – интерфејс који дефинише начине на које апликације могу да захтијевају услуге од библиотека, оперативних система или других апликација.

СПИСАК СЛИКА

Слика 1. Централизована и дистрибуирана мрежа....3

Слика 2. Ток трансакције у *blockchain* мрежи....4

Слика 3. Хеш показивач....7

Слика 4. Структура *blockchain* ланца....8

Слика 5. Стуктура Мерклеовог стабла....9

Слика 6. Трансакције организоване у форми Мерклеовог стабла унутар *blockchain*-а....10

Слика 7. Мале разлике улазних стрингова узрокују драстичне промјене излаза код *SHA-256* алгоритма....13

Слика 8. Примјер занемареног блока ван главног ланца....14

Слика 9. Ток извршавања паметних уговора....18

Слика 10. Пројекти унутар *Hyperledger*-а....20

Слика 11. Основни сервиси *Hyperledger Fabric*-а....22

Слика 12. Ток одобравања трансакције у *Hyperledger Fabric*-у....26

Слика 13. Кораци приликом одобравања трансакције у *Hyperledger Fabric*-у....26

Слика 14. Компоненте мреже специфициране *Hyperledger Composer* алатом....30

Слика 15. Ресурси и учесници моделовани унутар *blockchain* мреже за управљање ресурсима у видео играма....32

Слика 16. Трансакције моделоване унутар *blockchain* мреже за управљање ресурсима у видео играма....33

Слика 17. Унапријед креирани учесници у мрежи....35

Слика 18. Унапријед креирани *Game* ресурси....35

Слика 19. Унапријед креирани *ItemType* ресурси....36

Слика 20. Форма за иницирање трансакција и листа свих понуђених трансакција које се могу извршавати....37

Слика 21. Иницирање *BuyItem* трансакције....38

Слика 22. Нова инстанца ресурса *Item* креирана извршавањем *BuyItem* трансакције....38

Слика 23. Ново стање токена на рачуну играча....39

Слика 24. Јављање грешке приликом покушаја извршавања операције за који учесник нема право извршавања....40

Слика 25. Генерисане *REST* методе за ресурсе....41

Слика 26. Генерисане *REST* методе за учеснике....41

Слике 27. Генерисане *REST* методе за трансакције....41

Слике 28. Генерисане системске *REST* методе....42

Слика 29. Приказ *Game* ресурса....42

Слика 30. Форма за додавање новог *ItemType* ресурса....43

СПИСАК ТАБЕЛА

Табела 1. Особине јавних мрежа и мрежа са правима приступа.....6

Табела 2. Поређење најпознатијих алгоритама за постизање консензуса 16

СПИСАК ЛИСТИНГА

Листинг 1. Логика *BuyTokens* трансакције 48
Листинг 2. Логика *BuyItem* трансакције 49
Листинг 3. Логика *TradeItems* трансакције 50
Листинг 4. Логика *ListItem* трансакције 51
Листинг 5. Логика *Offer* трансакције 52
Листинг 6. Логика *CloseBidding* трансакције 53

БИОГРАФИЈА

Марина Ненић је рођена 11.01.1994. године у Брчком, Босна и Херцеговина. Завршила Гимназију "Филип Вишњић" у Бијељини. Факултет техничких наука у Новом Саду, смјер Рачунарство и аутоматика, усмјерење Примјењене рачунарске науке и информатика, уписала је школске 2012/2013. године. Добитник је стипендије Фонда за младе таленте. Основне академске студије завршила је школсе 2016/2017. године и исте године је уписала мастер студије на Факултету техничких наука. Положила је све испите прописане планом и програмом.