Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Aljaž Koželj

Razvoj decentraliziranih aplikacij na platformi Ethereum

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Matjaž Branko Jurič

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Preučite koncept in arhitekturo decentraliziranih aplikacij in jih primerjajte s klasičnimi aplikacijami. Podrobno preglejte razvoj decentraliziranih aplikacij na platformi Ethereum in opišite korake razvoja. Proučite koncept pametnih pogodb in področje decentraliziranih podatkovnih baz ter identificirajte ključne produkte. Opišite in razložite delovanje oraklov. Izdelajte praktični primer decentralizirane aplikacije, opišite korake razvoja in delovanje aplikacije ter postopek testiranja.

Zahvaljujem se svojemu mentorju prof. dr. Branku Matjažu Juriču za pomoč in vodenje pri pisanju diplomskega dela. Zahvalil bi se tudi staršem, sestri in prijateljem, ki so me pri tem spodbujali.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
dApp	decentralized application	decentralizirana aplikacija
EVM	Ethereum virtual machine	Ethereum virtualni stroj
KVS	key/value storage	shramba tipa ključ/vrednost
HTML	hypertext markup language	jezik za označevanje nadbesedila
CSS	cascading style sheets	kaskadne stilske podloge
HTTP	hypertext transfer protocol	protokol za prenos nadbesedila
IPC	inter-process communication	medprocesna komunikacija
API	application programming interface	aplikacijski programski vmesnik
ABCI	application blockchain interface	aplikacijski vmesnik za verigo blokov
DAO	decentralized autonomous organization	decentralizirana avtonomna organizacija
IPFS	interplanetary file system	interplanetarni datotečni sistem
JSON	JavaScript object notation	notacija za označevanje JavaScript objektov
BSON	binary JavaScript object notation	binarna notacija za označevanje JavaScript objektov
JSON-RPC	remote procedure call based on JavaScript object notation	oddaljen klic procedure baziran na notacije za označevanje JavaScript objektov

SHA3-256	secure hash algorithm 3 with 256 bit output	varen zgoščevalni algoritem 3 z rezultatom velikim 256 bitov
URL	uniform resource locator	enolični krajevnik vira
ABI	application binary interface	binarni vmesnik aplikacije
IDE	integrated development environment	integrirano razvojno okolje

Povzetek

Naslov: Razvoj decentraliziranih aplikacij na platformi Ethereum

Avtor: Aljaž Koželj

V diplomski nalogi smo opisali osnovne koncepte decentraliziranih aplikacij in decentraliziranega hranjenja podatkov. Proučili smo tehnologije Ethereum, BigchainDB in Oraclize. Primerjali smo decentralizirane aplikacije z modernimi oblačnimi aplikacijami ter decentralizirane podatkovne baze s klasičnimi podatkovnimi bazami. Ugotovili smo, da decentralizirane aplikacije in podatkovne baze omogočajo uporabniku večji nadzor pri uporabi in so bolj odporne na izpade, cenzuro ter regulacijo. Nadalje so decentralizirane aplikacije omejene pri svoji hitrosti izvajanja in kompleksnosti kode v poslovni logiki. To je posledica počasnih potrjevanj transakcij in cene izvajanja kode v verigi blokov. Omejitve imajo tudi pri pridobivanju podatkov izven verige blokov, saj koda v verigi blokov ne more neposredno dostopati do zunanjih podatkov. Identificirane omejitve je potrebno nasloviti pri razvoju decentraliziranih aplikacij, kar smo nazorno prikazali na praktičnem primeru.

Ključne besede: decentralizirana aplikacija, decentralizirana podatkovna baza, orakel, Ethereum, BigchainDB, Oraclize.

Abstract

Title: Decentralized application development on Ethereum platform

Author: Aljaž Koželj

In the diploma thesis we described the basic concepts of decentralized applications and decentralized data storage. We studied the Ethereum, BigchainDB and Oraclize technologies. In this work we carried out a detailed comparison between decentralized applications and cloud native applications, together with decentralized databases and classic databases. We discovered that decentralized applications and databases give the user more control, while being more resistant to faults, censorship and regulation. During our work, we realized that the current decentralized applications are limited at execution speed and code complexity. This is caused by the slow transaction confirmation and code execution costs on the blockchain. They also have a limited access to outside data, because the code inside the blockchain can't access external sources directly. The identified limitations must be addressed during the development of decentralized applications, which we showcase on a practical example.

Keywords: decentralized application, decentralized database, oracle, Ethereum, BigchainDB, Oraclize.

Poglavje 1

Uvod

1.1 Motivacija

Decentralizirane aplikacije so nova vrsta aplikacij, ki niso pod nadzorom centraliziranega akterja. Prva decentralizirana aplikacija je bila kriptovaluta Bitcoin, ki je tudi prva vpeljala verigo blokov. Veriga blokov je omogočila nadaljnji razvoj področja decentraliziranih aplikacij. To področje je v zadnjih letih doživelo eksponentno rast.

Danes je internet vse bolj centraliziran. Nadzoruje ga nekaj velikih podjetij. Zaradi tega je ideja decentraliziranega interneta zelo vabljiva. To je internet, kjer se nadzor od večjih podjetij prenese med njegove uporabnike. Razvoj in adaptacija decentraliziranih aplikacij je velik korak proti vpeljavi decentraliziranega interneta. Spletne aplikacije, ki so varne pred cenzuro in izpadi, zaupanje med razvijalci spletnih aplikacij in njihovimi uporabniki, način plačevanja brez omejitev držav in bank ter prenos nadzora interneta od velikih podjetij na njegove uporabnike. To so le nekatere od idej, ki še pred kratkim niso bile mogoče. S pomočjo decentraliziranih aplikacij te ideje postajajo realnost, proti kateri se premikamo. Razvijalci preko celega sveta so videli potencial te tehnologije in razvoj tega področja hitro raste.

1.2 Cilja

Kot glavni cilj diplomske naloge smo si zadali pregledati področji decentraliziranih aplikacij in decentraliziranega hranjenja podatkov. V delu smo preučili koncepte in tehnologije, ki omogočajo gradnjo decentraliziranega interneta.

Drugi cilj diplomske naloge je razvoj decentralizirane aplikacije, ki v svoji arhitekturi vključuje proučene tehnologije, in med razvojem identificirati njihove omejitve.

1.3 Struktura

V diplomskem delu najprej obravnavamo področje decentraliziranih aplikacij, kjer predstavimo osnovne koncepte, prednosti in tehnologije za razvoj. V naslednjem poglavju opišemo področje decentraliziranega hranjenja podatkov. Osredotočimo se na decentralizirane podatkovne baze in njihove koncepte. V nadaljevanju obravnavamo tematiko prenašanja podatkov v decentralizirane aplikacije. Proučimo koncept oraklov in njihovo vlogo v decentralizirani arhitekturi. Diplomsko delo zaključimo z razvojem decentralizirane aplikacije v kateri združimo proučene tehnologije.

Poglavje 2

Decentralizirane aplikacije (dApps)

Decentralizirane aplikacije so nova vrsta medmrežnih programov, ki so niso nadzorovani s strani centraliziranega akterja. Za svoje delovanje uporabljajo verigo blokov (angl. blockchain). To je porazdeljena glavna knjiga, ki je podvojena na večih vozliščih v omrežju. V primerjavi z klasičnimi aplikacijami so decentralizirane aplikacije veliko manj ranljive na izpade, cenzuro in regulacijo. Omogočajo prenos vrednosti, ki ga ne omejuje nobena banka ali država. Tehnologija decentraliziranih aplikacij je trenutno omejena pri hitrosti izvajanja in kompleksnosti kode.

2.1 Opis decentraliziranih aplikacij in osnovnih pojmov, ki jih opredeljujejo

Za decentralizirane aplikacije je značilno, da so odprtokodna programska oprema. Njihova koda mora biti dostopna vsem. Brez tega uporabniki ne morejo zaupati decentralizirani aplikaciji. Decentralizirana aplikacija mora tudi uporabiti kriptografsko valuto, s katero omogoči dostop do aplikacije. To valuto mora uporabiti tudi za nagrajevanje akterjev, ki prispevajo k delovanju aplikacije. Kriptovaluta mora biti generirana s strani decentralizirane

aplikacije s pomočjo kriptograskega algoritma. Noben akter v omrežju si ne sme lastiti večino njene kriptovalute. Vse spremembe v aplikaciji se morajo dogajati preko postopka sporazuma vozlišč v omrežju. Podatki in zapisi operacij aplikacije morajo biti kriptografsko varni in shranjeni v javni decentralizirani verigi blokov. S tem se prepreči možnost kritične točke odpovedi (angl. single point of failure).

Glavna platforma na področju decentraliziranih aplikacij je Ethereum. Še preden jo podrobno opišemo, pojasnimo še nekaj osnovnih pojmov, ki opredeljujejo decentralizirane aplikacije.

2.1.1 Osnovni pojmi

Decentralizirana aplikacija za svoje delovanje uporablja verigo blokov. V to verigo blokov se dodajajo bloki, ki vsebujejo transakcije. Bloki se v verigo blokov dodajo le v primeru soglasja dovolj udeležencev v omrežju. Akterji, ki sodelujejo pri procesu soglasja so lahko pri tem nagrajeni s kriptovaluto. Kriptovaluta se v glavnem uporablja kot plačilno sredstvo. Na nekaterih verigah blokov se pa lahko uporabi za izvajanje pametnih pogodb. Pametne pogodbe so programi, katerih koda se nahaja v verigi blokov.

2.1.1.1 Veriga blokov

Veriga blokov je porazdeljena glavna knjiga [4], ki je podvojena na večih vozliščih v omrežju. Svoje trenutno stanje hrani v obliki dnevnika transakcij [44]. Transakcija je osnovna enota verige blokov. Njen namen je opisati prenos vrednosti od ene osebe do druge. Veriga blokov omogoča nespremenljivo, pregledno in kriptografsko varno hranjenje zapisa transakcij. Transakcije se v omrežje dodajo le v primeru soglasja (angl. consensus) med udeleženci v omrežju [9].

Osebe so v verigi blokov predstavljene s pomočjo edinstvenega identifikatorja imenovanega naslov (angl. address). Kot naslov služi uporabnikov javni ključ [4]. Uporabnikov javni ključ je kriptografski ključ, ki se uporablja pri nesimetrični kriptografiji (angl. asymmetric cryptography). To je tip

kriptografije, ki omogoča uporabo javnega in zasebnega ključa za šifriranje in dešifriranje podatkov. Zasebni (angl. private) pozna samo lastnik. Javni (angl. public) ključ pa je znan vsem udeležencem omrežja [4].

V verigi blokov se kriptografska tehnologija uporablja za zagotavljanje integritete podatkov in za preverjanje legitimnosti izvedbe transakcije. Veljavne transakcije se zberejo v blok, ki se doda v verigo blokov. Poleg transakcij, ki se nahajajo v telesu bloka, lahko blok vsebuje dodatne informacije v svoji glavi. Sem se vključijo dodatne informacije, kot so časovni žig in kazalec na prejšni blok. V verigi blokov so bloki povezani tako, da ima vsak blok povezavo na svoj predhodni blok. Prvi blok v verigi blokov se imenuje genesis blok [4].

Postopek dodajanja nove transakcije v verigo blokov je sledeč [4]:

1. Vozlišče ustvari in digitalno podpiše transakcijo s svojim zasebnim ključem.

2. Transakcija je razposlana ostalim vozliščem v omrežju z namenom, da jo potrdijo. Za potrditev je potrebnih več vozlišč.

3. Potrjeno transakcijo se vključi v blok, ki se ga pošlje v omrežje. Transakcija velja za sprejeto.

4. Ustvarjen blok postane del glavne knjige. Veriga se podaljša za en blok.

Novi bloki se v verigo dodajo samo preko soglasja vseh vpletenih v omrežju. Pri tem uporabijo dogovorjen algoritem za soglasje. V primeru, da vozlišča niso prišla do soglasja, se blok zavrne [4].

2.1.1.2 Soglasje

Soglasje je proces doseganja konsenza med vozlišči v omrežju o končnem stanju podatkov v verigi blokov. S pomočjo algoritma za soglasje se omogoči decentralizacijo nadzora [44]. Algoritem za soglasje mora izpolnjevati naslednje pogoje, da pride do željenih rezultatov [4]:

- **Dogovor (angl. agreement)**: Vsa poštena vozlišča se odločijo za isto vrednost.
- **Prekinitev (angl. termination)**: Vsa poštena vozlišča zaključijo proces doseganja soglasja in na koncu dosežejo rešitev.
- **Veljavnost (angl. validity)**: Vrednost, za katero se odločijo vsa zaupanja vredna vozlišča, mora biti enaka vrednosti, ki jo je na začetku predlagalo vsaj eno vozlišče.
- **Odpornost na napake(angl. fault tolerant)**: Algoritem za soglasje mora delovati ob prisotnosti zlonamernih vozlišč.
- **Integriteta (angl. integrity)**: Vsako vozlišče lahko doseže največ eno odločitev znotraj enega postopka soglasja.

Obstaja več algoritmov za soglasje. Njihova uporaba je odvisna od potrebe verige blokov. Primera algoritmov za soglasje [4]:

- **Dokazilo o delu (angl. proof of work (PoW)):** Ta algoritem se zanaša na dokaz, da je bilo za dosego soglasja porabljenih dovolj računalniških virov. Zagotavlja, da je izvedba napada za napadalce predraga. Uporabljajo ga kriptovaluta Bitcoin in ostale kriptovalute.
- **Dokazilo o deležu (angl. proof of stake (PoS):** Pri tem algoritmu uporabnik rudari ali potrjuje transakcije glede na količino kriptovalute, ki jo ima v lasti. Več kriptovalute ima v lasti, več lahko rudari. Uporabniku z veliko kripto valute je v interesu, da ohrani omrežje varno [2]. Idejo je predstavil Peercoin. V prihodnosti bo nanj prešla platforma Ethereum.

Pri obeh naštetih algoritmih za soglasje igra poglavitno vlogo kriptovaluta. Akterji v omrežju, ki sodelujejo pri procesu soglasja, so ob uspehu nagrajeni z kriptovaluto decentralizirane aplikacije.

2.1.1.3 Kriptovaluta

Kriptovaluta je digitalno sredstvo, čigar stanje se hrani v verigi blokov. Veriga blokov za vsakega uporabnika hrani količino kriptovalute, ki mu pripada. Uporabniki, ki imajo v lasti kriptovaluto, jo lahko pošljejo drugemu uporabniku. Kriptovalute so bile ustvarjene z namen kompenziranja vozlišč v omrežju za proces soglasja. Pri procesu soglasja sodelujoči porabijo računalniške vire in morajo biti za uspešen proces soglasja nagrajeni [4].

Prva kriptovaluta je bila kriptovaluta Bitcoin. Ustvarjena je bila kot splošno plačilno sredstvo. Danes se večina kriptovalut uporablja kot plačilno sredstvo. Nekatere kriptovalute pa imajo drugačen namen. Primer take kriptovalute je kriptovaluta Ether. To je kriptovaluta v platformi Ethereum. Njen namen je uporaba pri plačevanju za izvajanje pametnih pogodb v platformi Ethereum[4].

2.1.1.4 Pametna pogodba

Pametne pogodbe so nespremenljivi programi, ki imajo zagotovljeno izvajanje[16]. Izvajajo v vsakem vozlišču. Vsaka decentralizirana aplikacija ima na verigi blokov eno ali več pametnih pogodb. Te pogodbe se uporabijo za poslovno logiko decentraliziranih aplikacij. [4].

Pogodbe se začnejo izvajati ob prejemu sporočil. Sporočilo vsebuje podatke, ki so potrebni za izvajanje funkcije. Skupaj s sporočilom se pošlje tudi kriptovaluta. Sporočila lahko pošljejo uporabniki ali druge pametne pogodobe. Koda v pametnih pogodbah lahko dostopa do podatkov v sporočilih. Dostopa lahko tudi do notranje hrambe podatkov. Vsaka pogodba ima svojo lastno notranjo hrambo podatkov.

Pametne pogodbe Ethereum so, tako kot ostali programi, napisane v programski kodi [31]. Najbolj popularen programski jezik za pisanje pametnih

pogodb Ethereum je programski jezik Solidity [17]. Spisano kodo pogodbe je najprej treba prevesti v bitno kodo in nato postaviti v omrežje Ethereum [4]. Ob postavitvi pogodbe se določiti kateri račun izvaja postavitev. Temu računu se zaračuna poraba plina, ki ga je potreboval postavitev pogodbe na omrežje [41].

2.1.2 Ethereum

Platforma Ethereum je decentralizirana platforma, ki poganja pametne pogodbe (angl. smart contract). Namen platforme Ethereum je vzpostavitev alternativnega protokola za izgradnjo decentraliziranih aplikacij. Z vpeljavo pametnih pogodb želijo razvijalcem prihraniti čas in ponuditi boljšo varnost.

Platforma Ethereum omogoča razvoj treh tipov aplikacij [18]:

- **Finančne aplikacije:** To so aplikacije, ki omogočajo uporabnikom nove načine upravljanja in uporabljanja kriptovalut. Sem sodijo kriptovalute, ki so narejen po standardu ETH20, kripto denarnice, investicijske pogodbe, itd. [18].

- **Pol-finančne aplikacije:** Te aplikacije uporabljajo kriptovalute, ampak uporaba kriptovalut ni njihov glavni namen. Imajo večjo nedenarno funkcionalnost [18]. Pri teh aplikacijah se ponavadi uporablja nek zunanji vir podatkov, do katerega aplikacija dostopa preko takoimenovanih oraklov (angl. oracle). Ti opravljajo vlogo posrednika med aplikacijo in napravami izven omrežja Ethereum [18].

- **Ne-finančne aplikacije:** Sem sodijo aplikacije, ki se ne ukvarjajo s kriptovalutami. Decentralizirane avtonomne organizacije (angl. Decentralized autonomous organization (DAO)) so ena vrsta teh aplikacij. Konceptno je DAO virtualna entita, ki ima določeno število članov ali delničarjev. Ti imajo pravico upravljanja z entito ob dovoljšni prisotnosti članov. Lastniki lahko z DAO upravljajo tako, da prenašajo sredstva ali spreminjajo kodo v pametni pogodbi. Spreminjanje kode

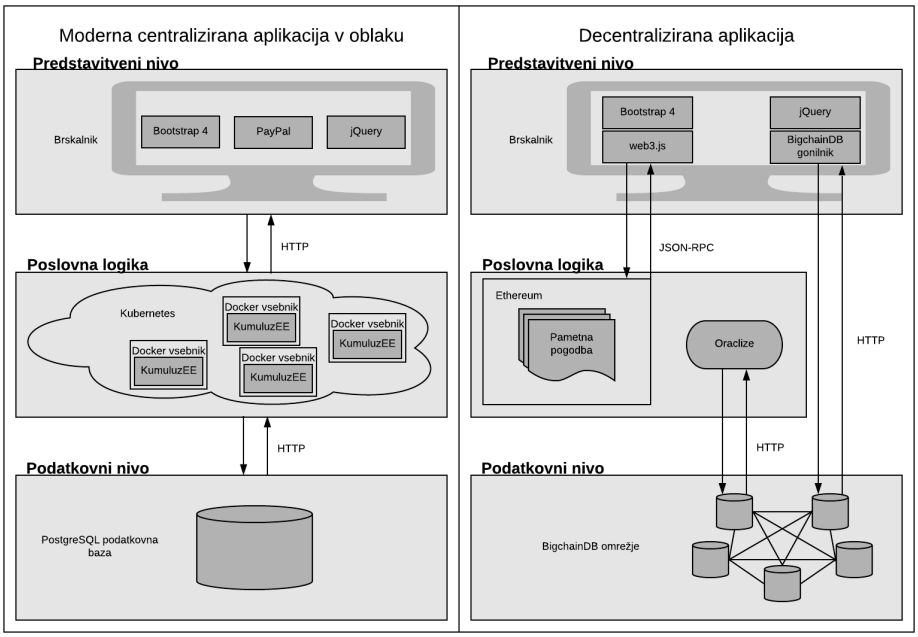
se lahko implementira z menjavo ciljne pametne pogodbe na kateri se nahaja poslovna logika. S tem se spremeni koda, ki se izvaja ob delovanju DAOja. Prvotna pametna pogodba ostane v verigi blokov, ampak je DAO ne kliče več [18].

Dodatni primeri decentraliziranih aplikacij Ethereum [18]:

- **Hranilnica**: Uporabnik lahko svoja sredstva prenese na pametno pogodbo, od koder jih lahko prenaša samo preko pravil v pametni pogodbi. S tem bi lahko preprečil preveliko dnevno porabo.
- **Zavarovanje**: Uporabnik lahko s pametno pogodbe sklene dogovor zavarovanja. Pametna pogodba bo na podlagi prejetih zunanjih virov in na podlagi sklenjenih pogojov uporabniku izplačala sredstva.
- **Decentraliziran vir podatkov**: Pogodba, ki služi kot vir podatkov ostalim pametnim pogodbam. Podatke priskrbijo uporabniki aplikacije, ki so za vnos pravilnih podatkov nagrajeni. Pravilen podatek se določi na podlagi vseh vnešenih podatkov.
- **Oblačne storitve**: Platforma Ethereum se lahko uporabi za ustvaritev preverljivega računskega okolja. To omogoča, da nek uporabnik kupi računsko moč pri drugem uporabniku. Ta uporabnik je, ob dokazilu izvedbe računskih operacij, plačan s strani kupca.
- **Igre na srečo**: Uporabniki, lahko stavijo na karakteristike naslednjega ustvarjenega bloka v verigi blokov Ethereum. S tem se prepreči vpliv zunanjega vira in omogoča poštene stave.

2.2 Primerjava s klasičnimi aplikacijami

Slika 2.1 prikazuje arhitekturi decentralizirane in klasične aplikacij. V nadaljevanju ju natančneje primerjamo.



Slika 2.1: Shema arhitekture moderne centralizirane aplikacije v oblaku (levo) in decentralizirane aplikacije (desno)

Iz tabele 2.1 je razvidno, da se klasične in decentralizirane aplikacije med seboj precej razlikujejo. Klasične aplikacije so z uporabo mikrostoritev v oblaku veliko bolj skalabilne in hitrejše od decentraliziranih aplikacij. Sle-

lastnost	Centralizirana	Decentralizirana
Skalabilnost:	velika	slaba
Hitrost:	hitra	počasna
Plačevanje:	potrebuje zunanjo storitev	vgrajeno
Gostovanje:	oblak	IPFS
Stroški izvajanja logike:	plača ponudnik aplikacije	plača uporabnik
Možnost izpada:	obstaja	zelo majhna

Tabela 2.1: Primerjava lastnosti klasičnih in decentraliziranih aplikacij

dnje imajo sicer na voljo bistveno več računalniških virov, a jih ne morejo izkoristiti tako kot klasične aplikacije. Vsako vozlišče mora obdelati vse transakcije, ki so sprejete v verigo blokov. V omrežju Ethereum je sodeluje na tisoče računalnikov, a transakcije se obdelujejo z efektivno hitrostjo enega samega.

Aplikacije se razlikujejo tudi v načinu gostovanja aplikacije. Klasične aplikacije se celotne gostijo v oblaku. Lastnik klasične aplikacije plačuje stroške in uporabniki imajo prost dostop. Pri decentraliziranih aplikacijah se vlogi zamenjata. Poslovna logika decentraliziranih aplikacij se izvaja v platformi Ethereum. Postavitev pogodbe stane razvijalca en enkratni znesek. Nadaljne stroške prevzamejo uporabniki aplikacije, ki plačujejo za njene funkcionalnosti. Ločeno se gosti uporabniški vmesnik.

Prednosti decentraliziranih aplikacijah so nahajajo pri primerjavi plačevanja in možnostih izpada. Možnost plačevanja že je vgrajena v platformo Ethereum. Klasične aplikacije morajo tu uporabiti storitev zunanjega ponudnika. Slednje tudi niso odporne na izpade, zaradi centralizirane infrastrukture. Decentralizirana aplikacija ima zelo majhno verjetnost izpada.

2.2.1 Predstavitveni nivo

Predstavitveni nivo predstavlja uporabniški vmesnik in je nivo s katerim ima uporabnik interkacijo. Uporabniku se v obliki spletne strani predstavi informacije, ki so bile pridobljene v komunikaciji z ostalimi nivoji [42].

Po izgledu in sami uporabi se predstavitvena nivoja klasičnih spletnih aplikacij in decentraliziranih aplikacij Ethereum ne razlikujeta veliko. Pri obeh se uporabljajo tehnologije HTML (jezik za označevanje nadbesedila (angl. hypertext markup language), CSS (kaskadne stilske podloge angl. (cascading style sheets)) in JavaScript za strukturiranje in delovanje spletnih strani [12].

Prva razlika se pojavi pri tem, da decentralizirane aplikacije Ethereum potrebujejo dodatno JavaScript knjižnico za povezovanje s poslovno logiko. Decentralizirane aplikacije namreč uporabljajo knjižnico web3.js za povezovanje in komuniniciranje z verigo blokov Ethereum. To knjižnico lahko priloži v spletno aplikacijo razvijalec sam ali pa jo priskrbi uporabnikov brskalnik. Slednja izbira je bolj pogosta in je mogoča s pomočjo brskalnika Ethereum. Primer takega brskalnika je brskalnik Mist. Mogoča je tudi uporaba vtičnika MetaMask, ki klasičen spletni brskalnik spremeni v brskalnik Ethereum. Brskalniki Ethereum so potrebni tudi za podpisovanje transakcij, kar je potrebno za delovanje decentralizirane aplikacije [30].

Decentralizirane aplikacije se od klasičnih razlikujejo tudi pri gostovanju uporabniškega vmesnika. Klasične aplikacije svoje uporabniške vmesnike gostijo na centraliziranih strežnikih v oblaku, ki si jih pogosto delijo s poslovno logiko. Ti strežniki uporabnikom posredujejo datoteke, v primeru dinamičnih spletnih strani pa lahko spletne strani tudi sproti generirajo [12]. Popolnoma decentralizirane spletne aplikacije svojih uporabniških vmesnikov ne gostijo na centralizirani infrastrukturi ampak pri decentraliziranem ponudniku. Za decentralizirano gostovanje se lahko uporabi decentraliziran datotečni sistem IPFS (interplanetarni datotečni sistem (angl. interplanetary file system)). S pomočjo decentraliziranega datotečnega sistema IPFS se lahko gosti statične spletne strani [12].

2.2.2 Nivo poslovne logike

Nivo poslovne logike je zadolžen za izvajanje logike spletne aplikacije. Obravnava in procesira zahteve uporabnikov. Izvaja komunikacijo s podatkovnim nivojem in uporabniškim vmesnikom. Uporabniškem vmesniku vrača odgovore na zahteve. [42].

Centralizirane spletne aplikacije poslovno logiko hranijo na centraliziranih strežnikih v oblaku. Tam se poslovna logika izvaja v obliki mikrostoritev, ki so nameščene v vsebnikih Docker. Decentralizirane aplikacije Ethereum se hranijo na verigi blokov v obliki pametnih pogodb, ki se izvajajo v EVM. Posledično je poslovna logika decentraliziranih aplikacij vedno dostopna, saj je možnost za izpad zelo majhna [12].

Poslovna logika centraliziranih aplikacij je implentirana v obliki mikrostoritev. To omogoča boljšo skalabilnost in je trenutno bistveno hitrejša od poslovne logike pri decentralizirani aplikaciji. Razlog za to je trenutna nezrelost tehnologij za decentralizacijo. Omrežje Ethereum trenutno omogoča 15 transakcij na sekundo [12], kar je bistvo počasnejše od klasičnih aplikacij (npr. VISA omogoča 45.000 transakcij na sekundo) [40].

Dodatna omejitev, ki je prisotna v poslovni logiki decentraliziranih aplikacij, je ta, da pametne pogodbe Ethereum niso zmožne samostojnega komuniciranja z napravami izven omrežja Ethereum. Pri poslovni logiki klasičnih aplikacij je veliko neposrednega komuniciranja s podatkovnim nivojem, pri decentralizirani pa to ni mogoče. Komunikacijo lahko decentralizirana poslovna logika izvede posredno preko oraklov [4].

Pri uporabi poslovne logike decentraliziranih aplikacij je treba poudariti to, da vsaka transakcija, ki je poslana v verigo blokov, uporabnika nekaj stane. Pri klasičnih aplikacijah stroške poslovne logika plača upravljalec aplikacije, pri decentraliziranih pa uporabnik. Razvijalec pri postavitvi pametne pogodbe v verigo blokov plača enkratni znesek, z nadaljnjim gostovanjem pa razvijalec oziroma lastnik nima dodatnih stroškov [12].

Pomembna razlika je tudi pri integraciji plačilnih storitev v spletno aplikacijo. Centralizirane aplikacije morajo za ponudbo teh storitev uporabiti

storitve ponudnikov kreditnih kartic ali bank in jih integrirati v svojo poslovno logiko. To lahko predstavlja dodaten razvojni čas in stroške. Platforma Ethereum že sama podpira uporabo plačevanja s svojo kriptovaluto. Uporaba tega v poslovni logiki je preprosta in prijazna razvijalcu [18].

2.2.3 Podatkovni nivo

Podatkovni nivo predstavlja repozitorij podatkov. Predstavitveni nivo in nivo poslovne logike oskrbuje s podatki [42].

Klasične centralizirane aplikacije za hranjenje podatkov uporabljajo relacijske in nerelacijske podatkovne baze. Te se nahajajo na centraliziranih strežnikih [42].

Hranjenje podatkov v decentraliziranem okolju dosežemo na dva načina:

- **Hranjenje v verigi blokov**: Platforma Ethereum omogoča, da vsaka pametna pogodba shrani svoje podatke kar na verigo blokov. Ti podatki so najbolj uporabni, če so shranjeni v obliki KVS. Tak način hranjenja je precej omejen, saj hranjenje večje količine podatkov postane drago. Hranjenje kompleksnejših strukturiranih podatkov tudi ni mogoče [21].

- **Hranjenje izven verige blokov**: Veliko bolj preprosto je hranjenje podatkov izven verige blokov. To se lahko izvede z uporabo decentraliziranih datotečnih sistemov, kot je decentraliziran datotečni sistem IPFS, ali decentraliziranih podatkovnih baz. Pametna pogodba Ethereum lahko do teh podatkov dostopa s pomočjo oraklov. Problem se pojavi pri procesiranju pridobljenih podatkov, saj ti ne smejo biti preveč kompleksni. Obdelovanje podatkov v platformi Ethereum je drago, zato je pogosta praksa, da pametna pogodba redko dostopa do zunanjih podatkovnih virov. Veliko bolj praktičen pristop je z referenciranjem datotek IPFS ali sredstev BigchainDB v pametni pogodbi, katera potem obdela logika na spletni strani. Dostop iz verige blokov do

takih podatkov naj se uporabi izključno pri podatkih, ki so ključnega pomena za izvajanje logike [27].

2.3 Platforma Ethereum in vse, kar se nanaša nanjo

Platforma Ethereum je odprtokodna globalna decentralizirana računalniška infrastruktura, ki izvaja programe imenovane pametne pogodbe. Pogosto ga enačijo z nazivom svetovnega računalnika. Za sinhronizacijo in hranjenje svojega globalnega stanja uporablja verigo blokov [4]. To stanje se v platformi Ethereum imenuje svetovno stanje (angl. world state) [4].

2.3.1 Virtualni stroj Ethereum

Del protokola Ethereum, ki se ukvarja z urejanjem sistemskega stanja in izvajanjem računskih operacij je virtualni stroj Ethereum (angl. Ethereum virtual machine (EVM)) [17]. EVM izvaja bitno kodo (angl. bytecode) pametnih pogodb v Turing-popolnem okolju. Kodo morajo izvesti vsa vozlišča v omrežju. S tem se sistemsko stanje spremeni v vseh vozliščih na enak način. Zaradi Turing-popolnosti je koda lahko poljubne zahtevnosti. EVM je popolnoma izoliran in nima dostopa do zunanjih virov. Izoliranost dodatno pripomore k varnosti [4]. Pri izvajanju virtualnega stroja Ethereum se porabi kriptovaluta Ether.

2.3.2 Kriptovaluta Ether

Virtualni stroj Ethereum za svoje izvajanje potrebuje računalniške vire vozlišč v omrežju. Platforma Ethereum zato nagradi vozlišča s svojo lastno kriptovaluto. Nagradi samo tista vozlišča, ki omrežju ponujajo svoje računalniške vire in sodelujejo pri procesu soglasja. S tem poplača porabljenje računalniške vire, ki jih vozlišča porabijo pri zagotavljanju varnosti v omrežju. Kriptovaluta se imenuje Ether in ima oznako ETH. Najmanjša denominacija krip-

tovalute Ether se imenuje Wei. Ena kriptovaluta Ether je enakovredna 10^{18} Wei. Kriptovaluta Ether je bila ustvarjena za plačevanje izvajanja pametnih pogodb v EVM. Uporabi se za nakup plina (angl. gas). Ta se porabi pri izvajanju računskih operacij v EVM. Količina porabljenega plina je odvisna od zahtevnosti izvajanja [4].

Za predstavitev lastništva in prenosa kriptovalute Ether se v verigi blokov Ethereum uporabljajo ključi in naslovi. Ključi so pari javnih in zasebnih ključev. Iz naključno generiranega zasebnega ključa se generira javni ključ. Iz javnega ključa se potem generira še naslov. Pri transakcijah se kriptovaluta Ether pošilja od enega naslova do drugega [4]. Pari javnih in zasebnih ključev se uporabljajo za kriptografsko podpisovanje transakcij in pošiljanje teh do računov.

2.3.3 Račun

Račun (angl. account) je eden od glavnih gradnikov v verigi blokov Ethereum. Vsak račun ima svoje stanje kriptovalute Ether in vzdržujejo svoj KVS (shramba tipa ključ/vrednost (angl. key/value store)) [4]. Iz praktičnega vidika je EVM en velik decentraliziran računalnik, ki vsebuje miljone račun (angl. account). Obstajata dva tipa računov [17]:

- **Računi v zunanji lasti (angl. externally owned account (EOA)):** Ta tip računa je nadzorovan preko zasebnega ključa. Lastnik zasebnega ključa računa v zunanji lasti lahko iz tega računa pošilja kriptovaluto Ether in sporočila [17]. V omrežju predstavljajo enega uporabnika [4].

- **Pogodbeni računi (angl. contract account (CA)):** To je račun, ki ima svojo lastno kodo in stanje. Izvajanje kode na računu se proži kot odziv na dobljena sporočila drugih računov [4].

Uporabniki lahko z računi upravljajo preko transakcij.

2.3.4 Transakcija

V platformi Ethereum je transakcija digitalno podpisan podatkovni paket. V transakciji se definira ciljni naslov, količina poslane kriptovalute Ether, zgornja meja dovoljene uporabe plina, digitalni podpis in dodatni podatki. Glede na vsebino se lahko transakcije deli na dva tipa [4]:

- **Transakcije, ki vsebujejo sporočilni klic (angl. message call transactions)**: Transakcija naredi sporočilni klic (angl. message call), ki se uporabi za prenos sporočila iz enega računa na drugega.
- **Transakcije, ki ustvarijo pogodbe (angl. contract creation transactions)**: Te transakcije povzročijo ustvaritev novega pogodbenega računa. Koda novega računa se pošlje skupaj s transakcijo. Ob uspešni transakciji se ta koda postavi v verigo blokov Ethereum.

Sporočilo je podatkovni paketek, ki se prenese med dvema računoma. Paketek vsebuje podatke in kriptovaluto Ether. Opiše tudi kdo je pošiljatelj in prejemnik tega sporočila. Sporočilo lahko pošlje uporabnik ali pametna pogodba. Pametna pogodba prejme sporočilo in ga prebere. Ob koncu izvajanja se sporočilo zavrže [4]. Izvedene operacije povzročijo spremembo stanja. Sprememba stanja poteka v naslednjih korakih [4]:

1. Preveri se pravilnost transakcije s pregledom sintakse in digitalnega podpisa.
2. Izračuna se cena transakcije in določi pošiljatelja. Preveri se stanje kriptovalute Ether na računu pošiljatelja.
3. Določena kriptovaluta se prenese iz pošiljateljevaga računa na ciljni račun. V primeru, da je ciljni račun pametna pogodba, se v tem koraku začne izvajanje kode ali postavitev nove pogodbe.
4. Napaka med izvajanjem povzroči prekinitev transakcije. Vsa stanja se ponastavijo na stanje pred začetkom izvajanja. Napako lahko pov-

zročijo pomanjkanje kriptovalute Ether na pošiljateljevem računu ali premajhna omejitev plina. Porabljen plin je plačan vozliščem.

5. Porabljen plin se plača vozliščom. Plin, ki se med izvajanjem ni porabil, se vrne pošiljatelju. Svetovno stanje preide v naslednje stanje.

Transakcije ustvarijo uporabniki izven verige blokov. Pri tem morajo uporabiti orodja, ki omogočajo gradnjo in podpisovanje transakcij Ethereum.

2.3.5 Povezovanje z verigo blokov Ethereum

Vsako vozlišče Ethereum v omrežju ima svoj vmesnik RPC. Preko tega vmesnika lahko spletna aplikacija dostopa do verige blokov Ethereum in vseh funkcionalnosti, ki jih ponuja vozlišče. Vmesnik je osnovan na specifikaciji za vmesnik JSON-RPC (oddaljen klic procedure baziran na notacije za označevanje JavaScript objektov (angl. remote procedure call based on JavaScript object notation)) 2.0, ki podpira serizalizacijo in je dostopen preko protokola HTTP (protokol za prenos nadbesedila (angl. hypertext transfer protocol)). Uporaba vmesnika JSON-RPC je precej zahtevna in nepregledna, zato se pripoča uporaba JavaScript knjižnice imenovane web3.js. Knjižnica uporabniku omogoča prijazno povezovanje na verigo blokov Ethereum [30].

Za obisk decentraliziranih aplikacij na spletu, uporabnik potrebuje brskalnik Ethereum. To je brskalnik, ki je sposoben interakcije z decentraliziranimi aplikacijami in hkrati deluje kot denarnica Ethereum (angl. wallet). Popularna sta brskalnik Mist in vtičnik Metamask za obstoječe brskalnike . Brskalniki Ethereum priskrbijo svojo različico knjižnice web3.js. Taka knjižnica ob transakcijah potem kliče denarnico brskalnika [4].

2.3.6 Izbira produkta

Za izgradnjo poslovne logike naše decentralizirane aplikacije smo izbrali platformo Ethereum. Zanjo smo se odločili, ker platforma Ethereum omogoča gradnjo močnih pametnih pogodb. Poleg tega je za pametne pogodbe Ethereum na voljo veliko dokumentacije in orodij, ki pomagajo pri razvoju.

Poglavje 3

Decentralizirane podatkovne baze

Tako klasične aplikacije kot tudi decentralizirane aplikacije Ethereum pri svojem delovanju uporabljajo shranjevanje in pridobivanje podatkov. Klasične aplikacije uporabljajo klasične podatkovne baze, kot so PostgreSQL ali MongoDB. Platforma Ethereum ima to funkcionalnost vgrajeno že v EVM. EVM omogoča shranjevanje stanja aplikacije v verigo blokov. Problem se pojavi pri shranjevanju večje količine podatkov, saj se za vsak porabljen bajt zaračuna plin. Cena transakcije hitro naraste zaradi velikosti podatkov. Problemu se je moč izogniti s tem, da se večje velikosti podatkov shranjuje izven verige blokov [27]. Ena od takih rešitev predstavljajo decentralizirane podatkovne baze.

3.1 Opis decentraliziranih podatkovnih baz

Decentralizirana podatkovna baza je tehnologija, ki ponudi funkcionalnosti klasičnih podatkovnih baz v decentraliziranem okolju [13]. Razvijajo se zaradi potrebe po decentraliziranem hranjenju večje količine strukturiranih podatkov. Decentraliziranim aplikacijam omogoča hranjenje večje količine podatkov. Omogoči jim tudi izvajanje poizvedb, ki so sicer značilne za klasične

podatkovne baze [21].

Omrežje decentralizirane podatkovne baze je sestavljeno iz mnogih vozlišč. Ta so razpšena po celem svetu, kar omogoči odpornost na izpade in cenzuro [21]. Vozlišča v omrežju lahko delujejo kot odjemalci (angl. consumer) ali ponudniki hrambe podatkov (angl. storage provider). Uporabniki najamejo prostor za svoje podatke in njihovi podatki so prenešeni do ponudnikov. Podatki se pri tem replicirajo in se shranijo na več različnih vozlišč v omrežju [13]. Uporabnik ima popoln nadzor nad svojimi podatki. Ima lastništvo in določi kdo ima pravice dostopa do njegovih podatkov. Shranjeni podatki so kriptografsko šifrirani, kar jih zaščiti pred krajo. [21].

3.2 Primerjava s klasičnimi bazami

V tabeli 3.1 je prikazana primerjava lastnosti klasičnih in decentraliziranih podatkovnih baz. V naslednjih podrazdelkih natančno primerjamo podatkovne baze glede na podane lastnosti.

lastnost	Centralizirana	Distribuirana	Decentralizirana
Upravljanje podatkov:	gostitelj	gostitelj	uporabnik
Nevarnost izpadov:	velika	mogoča	zelo majhna
Skalabilnost:	slaba	dobra	dobra

Tabela 3.1: Primerjava lastnosti podatkovnih baz

3.2.1 Upravljanje podatkov

Pri centraliziranih in distribuiranih podatkovnih bazah so podatki centralizirani na nekaj strežnikih, ki jih nadzoruje ponudnik storitve. Teh ponudnikov je na svetu le peščica, torej z večino podatkov upravlja le nekaj podjetij. Pri decentralizirani podatkovni bazi se podatki nahajajo na mnogih vozliščih v omrežju. Lastnik podatkov je edini, ki lahko upravlja s podatki. Podatki so tudi kriptografsko šifrirani, kar jih naredi neberljive za uporabnike, ki niso lastnik. Ti podatki so posledično odporni tako na cenzuriranje kot tudi krajo [21].

3.2.2 Nevarnost izpadov

Pri centralizirani podatkovni bazi se vsi podatki nahajajo na enem strežniku; eni napravi. Pri veliki količini zahtev po podatkih lahko naprava postane neodzivna, ker nima dovolj računalniških virov, da pravočasno obdela vse zahteve [29]. Strežnik lahko postane tudi žrtev zlonamernega napada, ki onemogoči njegovo delovanje. Možnost izpada pri takem tipu podatkovne baze je velika.

Distribuirana podatkovna baza je zbirka večih, logično povezanih podatkovnih baz, ki so porazdeljene preko omrežja. Težave centraliziranih podatkovnih baz se rešuje s pomočjo replikacije. Ob izpadu ene naprave iz omrežja, je podatek na voljo pri drugih napravah. Okvara enega vozlišča v omrežju ne povzroči izgubo podatkov in izpada celotnega sistema [29].

Pri decentralizirani podatkovni bazi je nevarnost izpadov rešena na podoben način kot pri distribuirani podatkovni bazi. Podatki, ki so poslani v omrežje decentralizirane podatkovne baze, so replicirani na več vozlišč v omrežju. Pri tem se zagotavlja minimalno število vozlišč na katerih se lahko nahaja en podatek. Ob izpadu enega izmed vozlišč se ti podatki po potrebi replicirajo na neko drugo vozlišče [13].

3.2.3 Skalabilnost

Skalabilnost pri centraliziranih podatkovnih bazah je mogoča z nadgradnjo računalniških virov. Ta je omejena zaradi finančnih in tehnoloških omejitev [29]. Centralizirane podatkovne baze so posledično slabo skalabilne.

Distribuirane podatkovne baze in decentralizirane podatkovne baze se skalirajo z dodajanjem novih vozlišč oziroma naprav v omrežje [29] [13]. Nova vozlišča pri distribuiranih podatkovnih bazah so novi strežniki ponudnika storitve [29]. Pri decentraliziranih podatkovnih bazah vozlišča niso nadzorovana s strani enega ponudnika, saj ne obstaja centralnega nadzora. Nova vozlišča predstavljajo nove akterje decentralizirane podatkovne baze, ki so se prostovoljno vključili v omrežje. Posledično so vozlišča pri decentra-

liziranih podatkovnih bazah bolj geografsko razširjena [13].

3.3 Produkti decentraliziranih podatkovnih baz

3.3.1 BigchainDB

Decentralizirana podatkovna baza BigchainDB je decentralizirana podatkovna baza, ki želi združiti distribuirano podatkovno bazo z značilnostmi verige blokov [34].

Zasnovana je kot distribuirana podatkovna baza, ki je bila nadgrajena s karakteristikami verige blokov: decentraliziran nadzor, nespremenljivost ter ustvaritev in premikanje digitalnih sredstev (angl. asset) [22].

Uporabna je povsod, kjer je potreba po nespremenljivih podatkih o digitalnih sredstvih in njihovem lastništvu. Namenjena je uporabi s strani tako decentraliziranih kot tudi centraliziranih aplikacij. Omogoča tudi zasebno rabo, kjer si korporacije postavijo svoje lastno omrežje BigchainDB [22].

3.3.2 Bluzelle

Decentralizirana podatkovna baza Bluzelle je ponudnik decentralizirane baze namenjen predvsem razvijalcem decentraliziranih aplikacij [6]. Da bi zagotovili najboljšo performanco, skalabilnost in zanesljivost, decentralizirana podatkovna baza Bluzelle implementira tehnologijo rojev (angl. swarm). Roj je večja skupina vozlišč, ki sodelujejo pri hranjenju in upravljanju s podatki [6].

Akterji v sistemu so odjemalci in ponudniki. Ponudnik je lahko kdor koli, ki ima na svojem računalniku odvečen prostor na disku. V zameno za hranjenje podatkov od odjemalca prejme plačilo v obliki kriptovalute [6]. Decentralizirana podatkovna baza Bluzelle upravlja z dvema kriptovalutama. Prvi je narejen po standardu ERC20 in se imenuje BLZ. Tega lahko lastnik zamenja za žeton BNT, ki se uporabi pri najemu prostora pri ponudniku [6].

3.3.3 Fluence

Decentralizirana podatkovna baza Fluence je decentralizirana podatkovna baza, ki omogoča hranjenje strukturiranih in kriptiranih podatkov ter izvajanje poizvedb. Trenutno je še v fazi razvoja [13].

Uporabljajo koncept pogodb (angl. contract), ki predstavlja dogovor med odjemalcem in ponudniki hranjenja podatkov. Odjemalci sklenejo pogodbo s ponudnikom pri katerem bodo hranili podatke. Ponudnik je pri tem nagrajen s kriptovaluto. Pogodobe so shranjene v verigi blokov preko vseh vozlišč v omrežju. Nespoštovanje pogodb ali zlonamerni poizkusi s strani ponudnika mu lahko povzroči kazen [13].

V omrežju Fluence so vozlišča organizirana v gruče (angl. cluster). Vsaka gruča je namenjena določeni zbirki odjemalcov. Faktor replikacije je v primerjavi z verigo blokov občutno manjši, kar omogoča učinkovito upravljanje večje količine podatkov. Gruče nadzirajo izbrani arbitri, ki rešujejo spore med gručami [13].

3.3.4 Izbira produkta

Decentralizirana podatkovna baza Fluence je še vedno v fazi razvoja in nima postavljenega testnega omrežja [21]. Decentralizirani podatkovni bazi Bluzelle in BigchainDB imata testno omrežje, kar omogoča njuno uporabo pri razvoju. Za dostop do omrežje obe zahtevata registracijo na spletni strani. Registracija na spletni strani od decentralizirane podatkovne baze BigchainDB je preprosta in uporabnik takoj dobi dostop. Dostop do decentralizirane podatkovne baze Bluezelle je otežen z registracijo, ki ni takoj potrjena in pogosto ne deluje. Za nadaljni pregled smo izbrali podatkovno bazo BigchainDB.

3.4 Pregled decentralizirane podatkovne baze BigchainDB

3.4.1 MongoDB

Vsako vozlišče v omrežju BigchainDB uporablja svojo instanco podatkovne baze MongoDB, v kateri hrani strukturirane podatke [22].

Podatkovna baza MongoDB je odprtokodna NoSQL dokumentna podatkovna baza. Ustvarjena je bila zaradi rastočih potreb po hranjenju podatkov. Podatke shranjuje v obliki zbirk (angl. collection). Vsaka zbirka vsebuje dokumente. Ti so shranjeni v binarni obliki JSON (notacija za označevanje objektov JavaScript (angl. JavaScript object notation)) formata, ki ga imenujejo BSON(binarna notacija za označevanje JavaScript objektov (angl. binary JavaScript object notation)) [3].

Ponuja tehniko, ki jo imenujejo Sharding. Sharding je postopek porazdelitve podatkov preko več vozlišč v omrežju MongoDB. To omogoča izvajanje večjega števila operacij nad podatki v podatkovni bazi [3].

3.4.2 Protokoli Tendermint

Decentralizirana podatkovna baza BigchainDB za mreženje in soglasje med vozlišči v svojem uporablja Tendermint. Vse komunikacije med vozlišči v omrežju BigchainDB potekajo s pomočjo protokolov Tendermint [22].

Protokoli Tendermint so programska oprema, ki omogoča varno in konsistentno replikacijo ene aplikacije na različnih napravah [35]. Tehnologija je odporna na bizatinske napake. Pod bizantinske napake se štejejo izpadi naprav v omrežju in zlonamerni napadi katere od neizpadlih naprav. Podpira bizantinske napake pri največ eni tretjini udeležencev v omrežju [35]. Zaradi odpornosti protokolov Tendermint na bizantinske napake je na ta tip napak odporna tudi programska oprema decentralizirane podatkovne baze BigchainDB [22].

Tendermint je sestavljen iz dveh komponent: Tendermint Core in ABCI

(aplikacijski vmesnik za verigo blokov (angl. application blockchain interface). Tendermint Core skrbi za soglasje med vozlišči v omrežju in za enako zgodovino transakcij na vseh napravah. ABCI je aplikacijski vmesnik, ki omogoča procesiranje transakcij v kateremkoli programskem jeziku [35].

3.4.3 Sredstvo BigchainDB

Sredstvo (angl. asset) v svetu verige blokov najbolj pogosto predstavlja digitalna sredstva ali kriptovalute. Za sredstva je značilno, da imajo lastnika, ki ima edini pravico prenesti lastništvo svojih sredstev na nekoga drugega. Primer takega sredstva sta naprimer kriptovaluti Bitcoin in Ether [22].

Sredstvo v decentralizirani podatkovni bazi BigchainDB pomeni reprezentacija kateregakoli fizičnega ali digitalnega predmeta. Predstavlja lahko lastništvo nad fizičnim predmetom ali kriptovaluto. V večini primerov verige blokov vsebuje en sam tip digitalnega sredstva. To je predvsem značilno za kriptovalute. Decentralizirana podatkovna baza BigchainDB podpira ustvaritev poljubnih sredstev [22]. Edina omejitev je ta, da se sredstvo lahko predstavi kot asociativen seznam [38].

Sredstva uporabnik ustvari s pomočjo transakcije, ki sredstvo ustvari in kot lastnika navede ustvarjalca [38]. Sredstvo potem živi v decentralizirani podatkovni bazi BigchainDB v neskončnost, saj je nespremenljiv (angl. immutable). Lastništvo sredstva se lahko razdeli med več uporabnikov. Vsak uporabnik dobi v last delnice sredstva. Takemu sredstvu se reče deljeno sredstvo. Število delnic se nastavi ob njegovi ustvaritvi.

3.4.4 Identita uporabnika

Decentralizirana podatkovna baza BigchainDB za določanje identite uporabnika znotraj omrežja uporablja sistem za podpisovanje z javnimi ključi Ed25519 (angl. Ed25519 public-key signature system) [15]. Ta sistem se uporablja za generiranje para javnega in zasebnega ključa, računanje podpisev in preverjanje podpisov. [38]

3.4.5 Transakcija BigchainDB

Ustvarjanje in prenosi sredstev znotraj decentralizirane podatkovne baze BigchainDB se izvajajo s pomočjo transakcij [39].

Obstajata dva tipa transakcij:

- **Ustvarjalna (angl. Create) transakcija:** Ustvarjalna transakcija se uporablja za ustvarjanje novega sredstva v decentralizirani podatkovni bazi BigchainDB. V tej transakciji se določi tudi število delnic, če se ustvari deljeno sredstvo [39].
- **Prenosna (angl. Transfer) transakcija:** Prenosna transakcija se uporabi za posodobitev ali prenos lastnišva nad sredstvom. Nanaša se lahko samo na eno sredstvo, oziroma njegove delnice [39].

Transakcija BigchainDB je niz JSON, ki je strukturiran v skladu s transakcijsko specifikacijo BigchainDB. Specifikacija določa pričakovane vrednosti posameznih polj JSON, kako zgraditi transakcijo, kako preveriti pravilnost njene strukture ter kako jo podpisati s svojim zasebnim ključem [22]. V transakciji se določi s katerim sredstvom se želi delati in kakšna operacija se izvede nad njim. Transakcija je najbolj osnoven tip podatka shranjen v podatkovni bazi BigchainDB [39].

```
{
  "id": "3667c0e5cbf1fd3398e375dc24f47206cc52d53d771ac68ce14d
  ↪ df0fde806a1c",
  "version": "2.0",
  "inputs": [
    {
      "fulfillment": "pGSAIEGwaKW1LibaZXx7_NZ5-V0alDLvrguGLyL
      ↪ RkgmKWG73gUBJ2WpnabOY-4i-kSGFa_VxxYCcctpT8D6s4uTGOO
      ↪ F-hVR2VbbxS35NiDrwUJXYCHSH2IALYUoUZ6529Qbe2g4G",
      "fulfills": null,
      "owners_before": [
        "5RRWzmZBKPM84o63dppAttCpXG3wqYqL5niwNS1XBFyY"
      ]
    }
  ],
  "outputs": [
    {
      "amount": "1",
      "condition": {
        "details": {
          "public_key":
          ↪ "5RRWzmZBKPM84o63dppAttCpXG3wqYqL5niwNS1XBFyY",
          "type": "ed25519-sha-256"
        },
        "uri": "ni:///sha-256;d-_huQ-eG-QQD-GAJpvrSsy7lLJqyNh
        ↪ tUAs_own7aTY?fpt=ed25519-sha-256&cost=131072"
      },
      "public_keys": [
        "5RRWzmZBKPM84o63dppAttCpXG3wqYqL5niwNS1XBFyY"
      ]
    }
  ],
  "operation": "CREATE",
  "asset": {
    "data": {
      "message": "Greetings from Berlin!"
    }
  },
  "metadata": null
}
```


Slika 3.1: Primer izpisa vsebine transakcije v formatu JSON

Komponente transakcije po vrsti iz slike 3.1:

- **Enolični identifikator transakcije (angl. id):** Enolični identifikator transakcije je zgoščena vrednost SHA3-256 (varen zgoščevalni algoritem 3 z rezultatom velikim 256 bitov (angl. secure hash algorithm 3 with 256 bit output) transakcije. Vzame se cel objekt JSON, pretvori v bajte in potem zgosti z zgoščevalno funkcijo SHA3-256. Rezultat je niz dolg 64 znakov [38].

- **Verzija (angl. version):** Verzija pove za katero transakcijsko specifikacijo BigchainDB je narejena ta transakcija [38].

- **Vhodi (angl. inputs) in izhodi (angl. outputs):** Transakcijski vhodi in izhodi so mehanizem s pomočjo katerega se prenese nadzor ali lastništvo nad sredstvom oziroma njegovim delnicam [38].

  Transakcijski vhod je, v konceptualnem smislu, kazalec na en izhod od obstoječih transakcij. Določa kdo je bil prejšni lastnik sredstva in zraven priloži dokaz, da so bili pogoji za prenos lastništva izpolnjeni [22].

  Transakcijski izhod določa pogoje, ki morajo biti izpolnjeni, da se prenese lastništvo nad nekim sredstvom. Najbolj osnoven pogoj, ki mora biti izpolnjen, je podpis z zasebnim ključem lastnika [22].

  Posamezna transakcija lahko vsebuje več izhodov, s čimer se ustvarijo deljena sredstva [22]. Pri tem je treba paziti, da je vsota števila sredstev preko vseh izhodov enaka številu sredstev v vhodu [39].

  Izhodi podpirajo tudi uporabo kompleksnih pogojev. Primer kompleksnega pogoja je zahteva po podpisu z javnim ključem več uporabnikov hkrati. Decentralizirana aplikacija BigchainDB za implementacijo kompleksnih pogojev uporablja specifikacijo kripto-pogojev (angl. crypto-conditions [37]) [38].

- **Vhod:**

  Polje "izpolnjuje" (angl. fullfils) določa na kateri izhod katere transakcije se nanaša ta vhod. Določi identifikator transakcije in indeks izhoda transakcije, ker je izhodov lahko več. Pri ustvarjalna transakciji je to polje prazno [38].

  Polje "prejšni lastniki" (angl. owners before) predstavlja seznam javnih ključev tega sredstva. V ustvarjalni transakciji predstavlja uporabnike, ki so izdajatelji tega sredstva. Pri prenosni transakciji more biti enak seznamu javnih ključev v izbranem izhodu [38].

  Polje "izpolnitev" je dokaz, da je bil izpolnjen pogoj, ki ga določa polje "izpolnjuje". Vrednost je zgoščeni niz celotne transakcije, ki je

bil podpisan s potrebnimi zasebnimi ključi [38].

- **Izhod:**

  Polje "pogoj" (angl. condition) določa pogoj, ki bo moral biti izpolnjen, če želi kdo uporabiti ta izhod, in njegov kriptografski URI. Decentralizirana podatkovna baza BigchainDB za to uporablja kripto-pogoje, ki definirajo dva tipa pogojev [38].

  Prvi tip se imenuje pogoj ED25519-SHA-256 in kot cilj se pri tem tipu nastavi en javni ključ. Pogoj izpolni zasebni ključ tega javnega ključa [38].

  Drugi tip se imenuje pogoj THRESHOLD-SHA-256 in omogoča dodatno logiko pri postavljanju ciljev pri pogoju. S poljema prag (angl. threshold) in podpogoji (angl. subconditions) določimo koliko podpogojev more biti izpolnjenih, da se izpolni celoten pogoj. Podpogoj je lahko obeh tipov. Zahteva enega podpogoja predstavlja ALI (angl. OR) operacijo, zahteva vseh pa IN (angl. AND) operacijo [38].

  Polje URI se generira po specifikacij kripto pogojev [37] [38].

- **Operacija (angl. operation):** Operacija določa tip transakcije. Možna tipa sta "CREATE" (ustvarjalna transakcija) in "TRANSFER" (prenosna transakcija) [38].

- **Sredstvo (angl. asset):** Sredstvo predstavlja podatke, ki jih shranimo v podatkovno bazo. Polje je lahko prazno [38]. Zaradi nespremenljivosti podatkov v podatkovni bazi, je treba podatke določiti samo pri ustvarjalni transakciji [22]. Pri prenosnih transakcijah je treba podati identifikator transakcije [38].

- **Metapodatki (angl. metadata):** Metapodatki so dodatni podatki, ki se nanašajo na transakcijo. Posodobijo se lahko z vsako transakcijo [22].

3.4.6 Razvoj

Decentralizirana podatkovna baza BigchainDB podpira uporabo tako javnih omrežij, kot tudi zasebno ustvarjenih omrežij. Koda za strežnike je odprtokodna in ne potrebuje licence. V razvojnem okolju se priporoča uporaba testnega omrežja BigchainDB Testnet [36], ki je namenjen gradnji aplikacij za dokazovanje koncepta (angl. proof of concept).

Pri razvoju aplikacij, ki uporabljajo decentralizirano podatkovno bazo BigchainDB, je največ dela z izgradnjo transakcij. Pri tem so v veliko pomoč gonilniki (angl. drivers) BigchainDB. V času pisanja sta uradno podprta gonilnika za programska jezika Python in JavaScript. Razvijalska skupnost BigchainDB ponuja dodatno podporo za jezike Haskell, Go, Java in Ruby [14].

Transakcijo se lahko zgradi tudi brez pomoči gonilnikov. Pri tem je treba paziti, da je zgrajena v skladu s transakcijsko specifikacijo BigchainDB [39]. Paziti je potrebno tudi na skladnost s specifikacijo kripto-pogojev [37] za izbrani programski jezik [38].

Koraki gradnje transakcije:

- Izberejo se pravilne vrednosti za verzijo, operacijo, sredstvo in metapodatke transakcije [38].
- Pridobijo se vsi javni ključi, ki so potrebni v tej transakciji. Sem sodijo ključi tako preteklih kot novih uporabnikov [38].
- Ustvari se seznam vseh izhodov [38].
- Ustvari se seznam vseh vhodov. Vhodi še nimajo izpolnjenega polja za izpolnitev [38].
- Iz zgornjih elementov se sestavi asociativen seznam s pravilnimi ključi. Polje za identifikator transakcije je prazno [38].
- Iz seznama se ustvari niz, ki je oblikovan po formatu JSON. Ta se bo kasneje uporabil pri kriptografskih operacijah [38].

- Za vsak vhod:
  - V primeru prenosne transakcije se nizu dodata vrednost identifikator transakcije ter indeks izhoda, ki ga izpolnjujemo [38].
  - Niz se pretvori v bajte in iz njega se izračuna zgoščena vrednost SHA3-256 [38].
  - Zgoščeno vrednost se kriptografsko podpiše s potrebnimi zasebnimi ključi. Pri tem se uporabi kripto-pogoje [37]. Izračunana vrednost je vrednost izpolnitve pri tem vhodu [38].
- V poljih za vnose nastavimo vrednosti izpolnitev [38].
- Z uporabo postopa zgoščevanja se iz asociativnega seznama izračuna identifikator transakcije. Ta se nastavi v asociativnem seznamu [38].

Transakcijo se nato pošlje na spletni vmesnik BigchainDB z uporabo protokola HTTP. Za to se lahko uporabi gonilnik ali katerokoli drugo orodje, ki podpira pošiljanje preko protokola HTTP. Naslov URL (enolični krajevnik vira (angl. uniform resource locator)) je odvisen od omrežja, ki se uporablja [23].

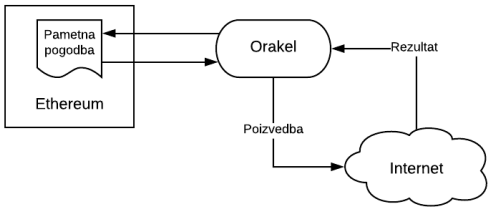
Poglavje 4

Prenos podatkov v dApps

Pametne pogodbe Ethereum se lahko pogovarjajo samo z akterji znotraj verige blokov. Viri izven verige blokov so tem pametnim pogodbam (in posledično decentraliziranim aplikacijam) nedosegljivi. Decentralizirane aplikacije, ki za svoje delovanje potrebujejo zunanje vire, težavo rešijo z uporabo oraklov (angl. oracle). Orakel je storitev, ki od pametne pogodbe dobi zahtevo po zunanjem viru in ji ga priskrbi. Pametna pogodba mora oraklu zaupati, da bo pridobil prave in resnične podatke.

4.1 Koncept oraklov

Orakli so pomemben člen ekosistema pametnih pogodb, katerih namen je pridobivanje zunanjih podatkov. Orakel si lahko predstavljamo kot vmesnik, ki



Slika 4.1: Shema koncepta oraklov

posreduje podatke med zunanjim virom podatkov in pametnimi pogodbami [4]. Slika 4.1 prikazuje vlogo orakla v decentralizirani aplikaciji Ethereum. Zunanje podatke v verigo blokov posredujejo s pomočjo transakcij [19], preko varnih kanalov. Sposobni so prenašanja poljubnega tipa podatkov [4].

Orakli ne smejo spreminjati poslanih podatkov. Avtentičnost pridobljenih podatkov dokažejo preko digitalnega podpisovanja. Digitalno podpisovanje ne zagotavlja, da so pridobljeni podatki pravilni, saj je lahko orakel napačne podatke dobil že od zunanjega vira. Uporabnik mora posledično zaupati tudi zunanjemu viru, saj ta lahko spremeni podatke brez vednosti svojih odjemalcev in lahko s tem povzroči zlorabo aplikacije. Decentralizirana aplikacija, ki mora slepo zaupati enemu centraliziranem viru, ni povsem decentralizirana [4]. Ena od rešitev je uporaba večih virov hkrati. Orakel ciljni podatek pridobi iz večih virov in se na podlagi dobljenih podatkov odloči kaj bo poslal uporabniku. S tem se odpravi nevarnost centralizacije zunanjega vira [4]. Omejitev se pojavi pri procesiranju podatkov iz različnih virov. Vsak vir lahko posreduje podatke v različni obliki. [28].

Problem centralizacije se pojavi tudi pri oraklu samem. Ti so v glavnem centralizirane storitve, ki živijo izven verige blokov. Zaradi decentralizacijskih zahtev so se pojavili tudi decentralizirani orakli. To so orakli zgrajeni na distribuiranih sistemih. Decentralizirani orakli omogočajo pridobivanja podatkov iz neke druge verige blokov, kar bi dodatno zagotovilo avtentičnost podatkov. Akterji v omrežju decentraliziranega orakla med seboj tekmujejo pri pridobivanju pravilnih podatkov [4].

Pri vseh omenjenih oraklih je še vedno prisotna možnost napake. Napako lahko naredi orakel ali pa je vir, od katerega podatke pridobiva orakel, napačen. Obstaja tudi možnost, da je podatek že pri svojem vnosu bil napačen. Nastane zahteva po zagotovilu, da je bil podatek pravilno ustvarjen in da se ni spremenil od nastanka. To težavo rešujejo strojni (angl. hardware) orakli. Strojni orakli so fizične naprave, ki podatke posredujejo iz realnega sveta. Uporabni so pri meritvah in v IoT (internet stvari (angl. internet of things)). Pri tem potrebujejo mehanizem, ki preprečuje vdore v fizično

napravo, Naprave morajo preko digitalnega podpisovanja morajo dokazati avtentičnost svojih podatkov. Taka rešitev je mogoča samo pri podatkih, ki jih je moč izmeriti z napravami [4].

4.2 Projekt Oraclize in sorodni rešitvi

4.2.1 Oraclize

Storitev Oraclize je vodilna orakel storitev za pametne pogodbe in aplikacije na verigi blokov, ki dnevno servira na tisoče zahtevkov. Na voljo je za uporabo na platformah Ethereum, Rootstock, R3 Corda in EOS [28].

Storitev Oraclize je centralizirana storitev [10], ki od uporabnikov ne zahteva slepega zaupanja. Ob pridobljenih podatkih zraven priloži dokument, ki ga imenujejo dokaz avtentičnosti (angl. authenticity proof). Ta dokazuje, da storitev Oraclize pridobljenih podatkov ni spreminjala [28].

Za uporabo storitve Oraclize v pametni pogodbi Ethereum se najprej pošlje transakcija na pametno pogodbo Oraclize. Pri transakciji se pošlje plačilo in pravilno strukturirana poizvedba v obliki niza. Na podlagi poizvedbe bo storitev Oraclize pridobil podatke in jih poslal nazaj na naslov pogodbe, ki je zahtevala storitev. Storitev Oraklize ustvari novo transakcijo, ko pridobi podatke iz zunanjega vira. Transakcijo pošlje do pogodbe, ki je naredila poizvedbo. Pri tem kliče vnaprej določeno funkcijo pametne v katero posreduje pridobljene podatke. Storitev Oraclize je v platformi Ethereum podprt na glavnem omrežju in tudi vseh testnih omrežjih [4].

Poizvedba je seznam parametrov, ki morajo biti ovrednoteni, da se izvede zahteva po viru podatkov. Prvi argument je glavni in obvezen. Določa tip vira podatkov v zahtevku. Preostali argumenti določajo podatke, ki jih pametna pogodba želi poslati zunanjemu viru [28].

Storitev Oraclize podpira sledeče tipe virov podatkov [28]:

- **URL:** Omogoča dostop do spletnih strani in HTTP APIjev.

- **WolframAlpha:** Omogoča dostop do računskega stroja (angl. computational engine) WolframAlpha.
- **IPFS:** Omogoča dostop do vsebine datoteke v decentraliziranem datotečnem sistemu IPFS.
- **naključno:** Vrne naključne bajte.
- **računanje:** Vrne rezultat poljubne računske operacije.

Pri procesiranju rezultatov zahtevka Oraclize omogoča uporabo JSON, XML, HTML in binarnih pretvornikov [28].

4.2.2 ChainLink

Storitev ChainLink je varen vmesnik na verigi blokov, ki omogoča pametnim pogodbam na različnih omrežji, da se povežejo z zunanjimi viri. Storitev je ponujena na platformah Ethereum, Bitcoin in Hyperledge [10].

Omrežje LINK je prvo decentralizirano orakel omrežje. Omogoča komurkoli, da varno zgradijo pametne pogodbe, ki imajo dostop do zunanjih podatkov. Omrežje v zameno za kriptovaluto pametnim pogodbam prinaša podatke [10].

Podatke priskrbijo vozlišča v decentraliziranem orakel omrežju. Ta imajo denarno spodbudo, da pridobijo najboljše možne podatke. Uporabniki imajo zagotovljeno pristnost podatkov, saj jih preveri več neodvisnih virov. Pri tem pomagajo vgrajeni sistem za ugled (angl. reputation system), sistem za ponudbe (angl. bidding system) in decentralizirano omrežje, ki se zanaša na neodvisne ponudnike orakel storitev. Slaba vozlišča v omrežju so kaznovana ali pa odstranjena, najboljša vozlišča pa so nagrajena in posledično imajo v omrežju največ zaslužka. S tem se spodbudi vozlišča, da vedno priskrbijo pravilne podatke [10].

Storitev ChainLink je decentralizirana, zaradi česar bo storitev vedno dostopna, saj nima posamezne točke izpada. Kvaliteta podatkov je zagotovljeno visoka, saj se ti črpajo iz večih virov hkrati. Koda, ki agregira

rezultate različnih vozlišč v omrežju je javno pregledna. Vsak lahko preveri kako natačno se izvaja [10]

Glavna prednost storitve ChainLink je v tem, da uporabniku ni treba zaupati samo enemu viru zunanjih podatkov. To je posledica tega, da je v omrežju več akterjev, kjer vsak preveri več možnih virov. To je tudi glavna razlika z ostalimi ponudniki.

4.2.3 Mobius

Protokol Mobius je omrežje, narejeno na verigi blokov, ki želi to tehnologijo verige blokov spraviti v popularno rabo (angl. mainstream). To želi storiti s tem, da razvijalcev ponudi možnost ustvaritve lastnih decentraliziranih aplikacij in sistema oraklov, ki bi povezal tok podatkov in vsakdanje aplikacije z verigo blokov [26]. Mobius ni ponudnik oraklov, ampak so ti del ponudbe na platformi [10].

Razvojna orodja in APIji omogočajo vsakemu razvijalcu programske opreme, da ustvari decentralizirano aplikacijo brez veliko učenja [26]. Protokol Mobius ponuja visoko skalabilnost, varnost in nizke cene transakcij. Na Mobiusovi tržnici decentraliziranih aplikacij se sprejemajo plačila s hitrostjo več kot tisoč transakcij na sekundo. To je bistveno hitreje kot omogoča omrežje Ethereum pri plačevanju s kriptovaluto Ether ali ERC20 žetoni. Transakcije so tudi bistveno cenejše, saj protokol Mobius lahko naredi več kot 400.000 transakcij za ceno enega dolarja [26]. V času pisanja je povprečna cena transakcije Ethereum stala 18 centov, z občasno rastjo na 1 dolar [11].

4.2.4 Izbira produkta

Za uporabo pri praktičnem primeru smo izbrali storitev Oraclize. Storitev ChainLink je še vedno v fazi razvoja in njena uporaba ni bila mogoča. V primerjavi s protokolom Mobius je storitev Oraclize lažja za uporabo s Ethereum pametnimi pogodbami, ponuja boljšo integracijo in ima več dokumentacije.

Poglavje 5

Praktičen primer

Drugi cilj diplomske naloge je bila ustvaritev decentralizirane aplikacije, ki bi predstavila uporabo opisanih tehnologij. Ustvarili smo decentralizirano aplikacijo, ki omogoča upravljanje s plačniškimi in neplačniškimi vstopnicami. Decentralizirano aplikacijo smo zgradili v platformi Ethereum. S tem smo prikazali izgradnjo decentraliziranih aplikacij Ethereum . Za delno hranjenje podatkov smo uporabili decentralizirano podatkovno bazo BigchainDB. Cilj je bil uporaba teh tehnologij v praktičnem primeru. Prikazali smo tudi trenutne omejitve decentraliziranih tehnologij.

5.1 Opredelitev problema

V okviru diplomske naloge smo ustvarili aplikacijo, s pomočjo katero demonstriramo uporabo raziskanih tehnologij. Domena aplikacije je spletna prodajalna in menjalnica vstopnic. Ob ustvaritvi aplikacije, administrator določi ime dogodka, začetno ceno vstopnic in količino vstopnic. Aplikacija definira dva tipa vstopnic.

Prvi tip vstopnic so plačniške vstopnice. Te lahko uporabniki kupijo preko svojih računov v verigi blokov Ethereum, če so karte še na voljo. Vsak račun Ethereum si lahko lasti le eno karto. Za to omejitev smo se odločili, ker v okviru te aplikacije ni bilo potrebe po implementaciji dodatne logike, ki bi

skrbela za nakup večih kart.

Drugi tip vstopnic so neplačniške vstopnice. Lastništvo teh je določeno preko uporabnikovega javnega ključa BigchainDB. Na začetku si vse karte lasti administrator, ki jih nato deli s preostalimi ljudmi. Namenjene so deljenju vstopnic s svojimi znancem. Nov lastnik vstopnice lahko svojo vstopnico prenese na drugo osebo.

Na ciljni strani aplikacije lahko uporabnik preveri svoje lastništvo nad obema tipoma vstopnic. To stori tako, da posreduje svoj račun Ethereum ali svoj javni ključ BigchainDB.

Ciljna stran uporabniku omogoča tudi nakup plačniške vstopnice ali prenos neplačniške vstopnice. Ob nakupu plačniške karte mora imeti dostop do spletne denarnice Ethereum, s pomočjo katere se podpiše transakcija. Pri prenosu neplačniške vstopnice ima na voljo svojo zalogo vstopnic poslati največ trem drugim osebam. To določi preko njihovih javnih ključev BigchainDB. Če želi obdržati del teh kart, mora sebe navesti kot enega izmed njih. Ob prenosu je potrebno posredovati svoj zasebni ključ BigchainDB.

Administrator lahko ob posredovanju svojega zasebnega ključa BigchainDB spremeni ceno plačljive vstopnice. Sprememba cene se bo v pametnimi pogodbi uveljavila v največ enem dnevu, saj pogodba po tej informaciji povpraša enkrat na dan. Razlog za to je cena povpraševanj v storitvi Oraclize.

Poleg ciljne strani obstaja tudi stran, na kateri se dokaže lastništvo nad obema tipoma vstopnic. Dokazilo lastništva nad plačljivo vstopnico se dokaže s pomočjo podpisovanja z denarnico Ethereum. Lastništvo neplačljive vstopnice se pa dokaže preko posredovanja tako javnega kot zasebnega ključa BigchainDB. Pri tem se vstopnica porabi.

5.1.1 Diagram primerov uporabe

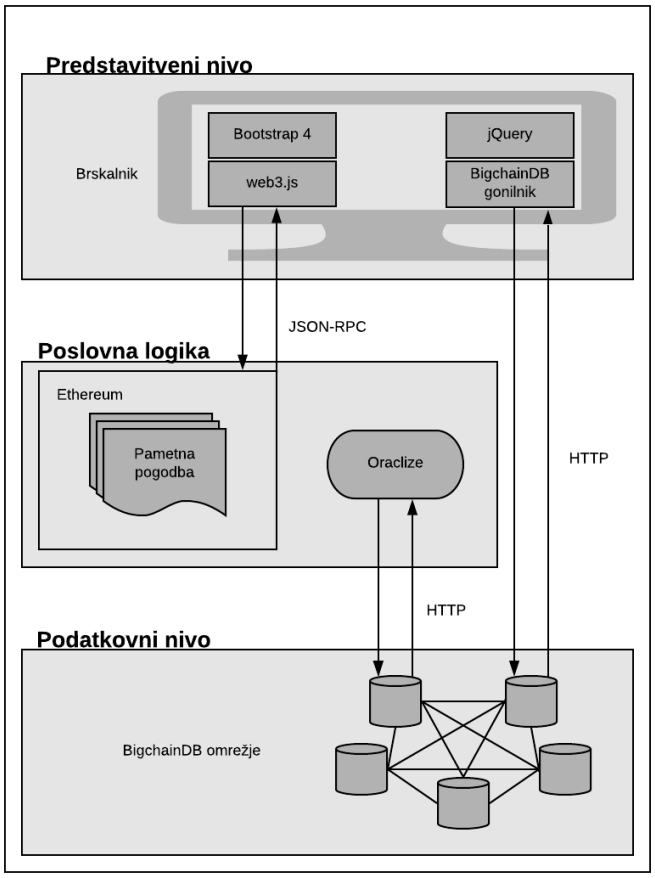
Slika 5.1 prikazuje diagram primerov uporabe. Predvidena sta dva tipa uporabnikov:

- **Uporabnik**: Uporabnik lahko v aplikaciji kupi plačniške vstopnice, prenese svoje neplačniške vstopnice ter preveri in dokaže lastništvo obeh tipov vstopnic.
- **Administrator**: Administrator lahko inicializira spletno prodajalno in menjalnico vstopnic ter nastavi ceno plačniških vstopnic. Ima tudi vlogo uporabnika in ima dostop do vseh njegov pravic.

Slika 5.1: Diagram primera uporabe

5.2 Arhitektura aplikacije

Arhitekturo naše decentralizirane aplikacije smo prikazali na sliki 5.2.



Slika 5.2: Shema arhitekture aplikacije

5.2.1 Podatkovni nivo

Na podatkovnem nivoju se nahaja decentralizirana podatkovna baza BigchainDB. Ta omogoči decentralizirano ustvaritev in nadzor nad digitalnimi sredstvi [22]. V decentralizirani aplikaciji je uporabljena za upravljanje sledečih sredstev:

- Sredstvo, ki opisuje dogodek. To je deljiv tip sredstva. Vsak delež predstavlja eno neplačniško vstopnico za dogodek.

- Sredstvo, ki opisuje ceno plačniške vstopnice za dogodek. Upravljalec tega sredstva je administrator.

5.2.2 Poslovna logika

Na nivoju poslovne logike se nahaja pametna pogodba Ethereum. Njena naloga je obravnavati zahteve po nakupu plačniških vstopnic in poizvedbe o njihovem lastništvu. Pogodba je vezana na digitalni sredstvi BigchainDB, ki opisujeta dogodek in ceno vstopnic.

Pametna pogodba ima tudi nalogo, da preverja morebitne spremembe o ceni vstopnice. Ta informacija se nahaja v decentralizirani podatkovni bazi BigchainDB v podatkovnem nivoju. Pogodba ni sposobna komunicirati z napravami izven omrežja Ethereum [19] in mora uporabiti orakel storitev, kot je prikazano na sliki 5.2.

Storitev Oraclize ponuja dobro integracijo z decentraliziranimi aplikacijami Ethereum in omogoča razčlenjevanje formata JSON. Pri tem je treba paziti, da pametna pogodba razširja pametno pogodbo usingOraclize, drugače pogodba nima dostopa do funkcionalnosti storitve Oraclize [28]. Naloga storitve Oraclize je, da enkrat na dan iz decentralizirane podatkovne baze BigchainDB pridobi informacijo o ceni vstopnice. Ta informacija se shrani v pametni pogodbi.

5.2.3 Predstavitveni nivo

Predstavitveni nivo je predstavljen s spletno aplikacijo. Naloga spletne aplikacije je prikazovanje uporabniškega vmesnika in obdelava uporabnikovih zahtev. To je edini del aplikacije, s katerim ima uporabnik interakcijo.

Spletna aplikacija se povezuje s pametno pogodbo na nivoju poslovne logike preko knjižnice web3.js. V pametno pogodbo posreduje zahteve uporabnika po nakupu in preverjanju lastništva plačniške vstopnice. Ob upravljanju z neplačniškimi vstopnicami se spletna aplikacija neposredno poveže na podatkovni nivo kot je prikazano na sliki 5.2. Z digitalnimi sredstvi znotraj decentralizirane podatkovne baze BigchainDB upravlja s pomočjo Javascript gonilnika BigchainDB.

Uporabniški vmesnik je zgrajen s pomočjo osnovnih spletnih tehnologij HTML, CSS in JavaScript. Pri tem se uporabijo tudi dodatne tehnologije, ki olajšajo razvoj in omogočijo dodatne funkcionalnosti.

Uporabljene tehnologije na predstavitvenem nivoju:

- **jQuery:** jQuery je hitra, majhna in funkcionalno-polna Javascript knjižnica. Omogoča obdelavo in manipulacijo HTML dokumentov ter delo z animacijami, dogodki in asinhronimi operacijami. Podpira uporabo na večih brskalnikih.

  jQuery je v spletni aplikaciji uporabljen za dinamično prikazovanje elementov na spletni strani. Prikaz je odvisn od interakcije uporabnika.

- **Boostrap:** Bootstrap je odprtokodno orodje, ki je namenjeno razvoju s HTML, CSS in JavaScript tehnologijami. Omogoča hitro prototipiranje spletnih strani s pomočjo vnaprej zgrajenih komponent in uporabo jQuery-ja.

  V spletni aplikaciji so uporabljene Boostrap componente za karto, gumbe in vnosna polja.

- **web3.js:** web3.js je JavaScript knjižnica, ki implementira generično implementacijo vmesnika JSON RPC [30]. Omogoča interakcijo z vo-

zlišči Ethereum preko protokola HTTP ali povezave IPC (medprocesna komunikacija (angl. inter-process communication)). Knjižnico vključi razvijalec sam ali pa jo priskrbi brskalnik Ethereum [30]. Primer takega brskalnika je vtičnik MetaMask. Prednost pri tej izbiri je ta, da vtičnik MetaMask omogoči dostop do računov Ethereum iz svoje denarnice [25].

- **Javascript gonilnik BigchainDB:** Gonilnik omogoča izgradnjo in izpolnjevanje transakcij BigchainDB, ter pošiljanje teh v omrežje [14].

5.3 Podrobnosti razvoja

Pred začetkom razvoja se je ustvarilo repozitorij Git. Orodje Git je odprtokodno orodje za nadzor nad verzijami datotek. V repozitorij[1] so se shranjevale vse verzije datotek. S tem se je preprečila morebitna izguba pomembnih datotek. Repozitorij se nahaja na oddaljenem spletnem strežniku ponudnika Github.

5.3.1 Razvoj pametne pogodbe

Razvoj rešitve za nakup vstopnic se je začel z razvojem pametne pogodbe. Pametno pogodbo smo razvijali v razvojnem okolju Remix. Razvojno okolje Remix je odprtokodno orodje, ki omogoča razvoj pametnih pogodb Solidity v brskalniku [33].

Za pisanje kode Solidity smo uporabili integrirano razvojno okolje (angl. integrated development environment (IDE)) Remix. Integrirano razvojno okolje Remix smo uporabili tudi za prevajanje kode in postavljanje prevedene kode v verigo blokov Ethereum. Pri postavljanju pogodbe v verigo blokov ima razvijalec možnost pošiljanja kriptovalute Ether ob postavitvi pogodbe. To bo začetno stanje kriptovalute Ether na tej pametni pogodbi.

Pogodba se postavi preko transakcije, ki je stane plin. Zaradi tega je potrebno transakcije podpisati z zasebnim ključem. V ta namen smo upo-

rabili vtičnik MetaMask. Vtičnik MetaMask spremeni brskalnik Chrome v brskalnik Ethereum, saj v vsako spletno stran injicira svojo knjižnico web3.js. MetaMask služi tudi kot denarnica Ethereum in omogoča ustvaritev računov Ethereum [25].

Celoten razvoj smo izvajali na testnem omrežju Ropsten [36]. Testno kriptovaluto Ether smo pridobili s pomočjo pipe Ethereum za testno omrežje Ropsten. Na tej spletni strani se vnese naslov uporabnikovega računa Ethereum, ki ga ima uporabnik v testnem omrežju Ropsten. Ta spletna stran na podan račun Ethereum prenese kriptovaluto Ether [20].

5.3.2 Razvoj spletne aplikacije

Spletno aplikacijo smo razvijali v urejevalniku kode Visual Studio Code. Hitro prototipiranje spletnih strani smo izvajali s pomočjo knjižnic Bootstrap [7] in jQuery [24]. V pomoč nam je bila spletna stran z Boostrap dokumentacijo in primeri.

V okviru spletne aplikacije smo razvili tudi vso logiko, ki upravljala s sredstvi BigchainDB. Pri tem smo uporabili JavaScript gonilnik BigchainDB [14] in testno omrežje BigchainDB. Dostop do testnega omrežja smo pridobili na spletni strani [5].

Na koncu je bilo potrebno spletno aplikacijo povezati s pametno pogodbo Ethereum. V ta namen smo uporabili knjižnico web3.js, katero priskrbi vtičnik MetaMask, in vmesnik ABI (binarni vmesnik aplikacije (angl. application binary interface)), ki se je generiral pri prevajanju kode v razvojnem okolju Remixu. Vmesnik ABI definira katere funkcije v pametni pogodbi lahko kliče knjižnica web3.js. Ta vmesnik smo v aplikacijo vključili preko njemu namenjene datoteke, ki jo uporabijo ostale datoteke JavaScript.

Kot zanimivost pri razvoju je treba omenit, da vtičnik MetaMask zahteva, da se knjižnico web3.js, ki jo injicira, uporablja samo na straneh, ki jih gosti spletni strežnik [25]. Rešitev nam je ponudila uporaba Pythonovega SimpleHTTPServerja, ki omogoča postavitev strežnika brez konfiguracije z enim ukazom. Strežnik se je pognal v mapi z vsemi datotekami. Tam poišče

datoteko imenovano "index.html", ki potem služi kot korenska stran strežnika [32].

5.4 Opis delovanja

5.4.1 Postavitev pametne pogodbe in inicializacija dogodka

Prvi korak pri postavitvi celotne aplikacije je postavitev pametne pogodbe. Pametno pogodbo se postavi s pomočjo spletnega razvojnega okolja Remix. Po prevodu kode, je potrebno nastaviti količino kriptovalute Ether, ki se bo poslala zraven ob postavitvi. Ta količina se bo čez čas porabila za proženje poizvedb v storitvi Oraclize. Administrator, ki se zaveda cene dnevne poizvedbe Oraclize, bo posredovano vrednost nastavil za nekaj dni. Od tam naprej bo dobiček od prodanih vstopnic kril nadaljne poizvedbe. Po nastavljeni vrednosti se v razvojnem okolju Remix sproži postavitev pametne pogodbe. Pri tem je treba transakcijo, ki vsebuje postavitev, podpisati z denarnico Ethereum.

Pri sveže postavljeni pametni pogodbi, je potrebno najprej inicializirati dogodek. V kolikor dogodek ni inicializiran, bodo vse zahteve na spletni strani preusmerjene na stran, kjer se dogodek inicializira. Dogodek inicializira uporabnik, ki je postavil pametno pogodbo.

Na strani za inicializaciji je potrebno izpolnit vsa vnosna polja:

- Ime dogodka
- Cena plačniške vstopnice (v wei)
- Število plačniških vstopnic
- Število neplačniških vstopnic
- Javni ključ BigchainDB

- Zasebni ključ BigchainDB

Inicializacija se sproži, ko uporabnik izpolni vse polja in pritisne na gumb za potrditev. Najprej se s pomočjo gonilnika BigchainDB ustvarita sredstvi za dogodek in za ceno dogodka. Pri sredstvu za dogodek se število deležev nastavi na število vseh neplačniških vstopnic. Pogoj za izpolnitev izhodov pri obeh sredstvih je zasebni ključ BigchainDB, ki ga je priskrbel uporabnik. S tem se določi začetnega lastnika vseh neplačniških vstopnic ter administratorja, ki bo upravljal s ceno plačniških vstopnic.

Ob uspešnem ustvaritvi obeh sredstev, se njuna identifikatorja in število plačniških vstopnic pošljejo na funkcijo pametne pogodbe, ki bo inicializirala decentralizirano aplikacijo s pravimi podatki. Ti se v pametni pogodbi shranijo, ob tem se tudi ustvari poizvedni niz za storitev Oraclize. Niz je strukturiran tako, da pridobi informacijo o ceni vstopnic iz metapodatkov iz najnovejše transakcije od sredstva BigchainDB s ceno. V sklopu inicializacije se tudi prvič proži poizvedba Oraclize. Cena te poizvedbe je 0 plina, ker je to prva poizvedba Oraclize iz te pametne pogodbe [28]. Preden se transakcija izvede, je potreben podpis z denarnico Ethereum.

V kolikor se transakcija Ethereum, ki inicializira aplikacijo, uspešno zaključi, ostale funcionalnosti aplikacije postanejo na voljo ostalim uporabnikom. Izjema je funckionalnost nakupa plačniške vstopnice, ki mora počakati na rezultat prve Oraclize poizvedbe, preden postane uporabna.

5.4.2 Pregled lastništva

Ob uspešni inicializaciji aplikacije, je uporabnik preusmerjen na ciljno stran. Tam ga pričakata dve vnosni polji, ki ju lahko izpolni:

- Polje za javni ključ BigchainDB
- Polje za naslov računa Ethereum

Ob izpolnitvi javnega ključa BigchainDB, se uporabniku prikažejo vse vstopnice, pri katerih ima njegov javni ključ lastništvo. Te so predstavljene

v obliki izhodov, ki so na voljo uporabniku. Ob tem se izpiše tudi količina vstopnic, ki so prepisane izhodu.

V primeru, da je bilo posredovan javni ključ administratorja za ceno vstopnic, se pokaže dodatno polje, kjer ima uporabnik možnost spremembe cene vstopnic.

Ob izpolnitvi naslova računa Ethereum se uporabniku prikaže možnost nakupa vstopnice oziroma sporočilo, da si uporabnik vstopnico že lasti. Pri tem se preko knjižnice web3.js dostopa do pametne pogodbe, kjer jo povpraša po ceni vstopnice in številu preostalih vstopnic. Povpraša tudi, če si priložen naslov Ethereum že lasti vstopnico. Uporabnik ima možnost nakupa vstopnice, če je še nima v svoji lasti in niso bile še vse prodane.

5.4.3 Prenos neplačniških vstopnic

Uporabnik, ki je lastnik vsaj ene neplačniške vstopnice, ima možnost prenesti lastništvo te vstopnice na nekoga drugega.

Ob izbiri enega izhoda, se mu odpre obrazec, ki ga mora izpolniti za prenos. Izpolniti mora polja za:

- Svoj zasebni ključ BigchainDB.
- Število novih izhodov. Število je omejeno na 3, zaradi lažjega razvoja.
- Javne ključe BigchainDB novih lastnikov. Število teh je odvisno od izbire števila izhodov.
- Število vstopnic pri vsakem novem izhodu. Vsota vseh mora biti enaka količini vstopnic v izhodu, ki ga izpolnjujemo.

Ob potrditvi se najprej pridobita transakcija in izhod, ki ga želimo izpolniti. S pomočjo izpolnjenih polj in pridobljenih transakcije in izhoda, se sestavi transakcija BigchainDB, ki prenese lastništvo na nove lastnike. Njeni izhodi so preslikave izpolnjenih polj v obrazcu. Podpisano transakcijo se

pošlje v omrežje BigchainDB. Ob potrditvi transakcije, se osveži uporabniški vmesnik in uporabnik ne vidi več izhoda, ki ga je pravkar izpolnil.

V kolikor lastnik hoče ostati lastnik dela vstopnic, se mora nastaviti kot novi lastnik s tem, da v obrazec zapiše tudi svoj javni ključ.

5.4.4 Nakup plačniške vstopnice

Uporabnik, ki ima omogočen nakup vstopnice, se lahko odloči za nakup lete. Ob potrditvi nakupa, se ustvari transakcijski objekt, ki potrebuje sledeče informacije:

- Naslov Ethereum, ki kupuje vstopnice.
- Vrednost, ki jo bo uporabnik poslal v pogodbo. To predstavlja ceno vstopnice.
- Plin, ki bo potreben za izvedbo transakcije.

Ta objekt se posreduje ob klicu funkcije za nakup vstopnice na pametni pogodbi. Za to se uporabi knjižnico web3.js, ki sproži zahtevo po podpisu transakcije z denarnico Ethereum.

Funkcija za nakup na pametni pogodbi najprej še enkrat preveri, če je nakup možen, in če je bila poslana zadostna vrednost. V primeru, da je bil klic funckije ustrezen se zmanjša število možnih vstopnic in kupcu pripiše, da si lasti vstopnico. Uporabniku se ob uspešni transakciji na uporabniškem vmesniku sporoči, da je bil nakup uspešno opravljen in, da si lasti vstopnico.

5.4.5 Sprememba cene vstopnic

Ta možnost se pojavi le osebi, ki je zadolžena za spremembo cene. Za spremembo cene mora vpisati svoj zasebni ključ BigchainDB in novo ceno vstopnice (v wei).

Ob potrditvi, se najprej pridobi indentifikator za sredstvo, ki opisuje ceno vstopnice. Identifikator je shranjen kot javna spremenljivka v pametni pogodbi. Za dostop do spremenljivke se uporabi knjižnica web3.js. S pomočjo

identifikatorja in gonilnika BigchainDB se pridobi najnovejšo transakcijo in izhod, ki mora biti izpolnjen. Iz izhoda in transakcije se ustvari nova transakcija, ki bo spremenila ceno vstopnice. Zaradi nespremenljivosti sredstev v decentralizirani podatkovni bazi BigchainDB se ceno shrani v metapodatkih.

5.4.6 Dokaz lastništva neplačniške vstopnice

Pri dokazovanju lastništva neplačniške vstopnice mora uporabnik posredovati svoj par javnega in zasebnega ključa BigchainDB. Dokazati je treba, da je uporabnik lastnik posredovanega javnega ključa BigchanDB.

Ob potrditvi se najprej iz pametne pogodbe pridobi identifikator sredstva, ki opisuje dogodek. Nato se pridobi prvi izhod, ki ustreza priloženemu javnemu ključu. Ustvari se nova transakcija, ki porabi izbrani izhod. Transakcijo se podpiše s priloženim zasebnim ključem in pošlje v omrežje BigchainDB. Pri tem se ponovno uporabi gonilnik BigchainDB. V primeru, da se transakcija uspešno izvede, je bilo dokazano lastništvo nad vstopnico.

Neplačniške vstopnice so prenosljive, zato je treba preprečiti, da se ista vstopnica uporabi večkrat. Decentralizirana podatkovna baza BigchainDB je nespremenljiva, zato ni mogoče zbrisati deleža sredstva. Zato se pri transakciji, ki je uporabljena za dokazovanje lastnišva, generira nov javni ključ BigchainDB, ki postane pogoj izhoda. Za generiranje ključa se uporabi gonilnik BigchainDB. Generirani javni ključ BigchainDB je naključen, kar zagotavlja izredno majhno možnost, da kdo najde njegov zasebni ključ BigchainDB. Vstopnica ni zbrisana, ampak je njen prenos praktično nemogoč [8].

5.4.7 Dokaz lastništva plačniške vstopnice

Pri dokazovanju lastništva plačniške vstopnice je treba posredovati naslov Ethereum.

Najprej je treba dokazati, da je uporabnik lastnik posredovanega naslova Ethereum. S pomočjo knjižnice web3.js se najprej kriptografsko podpiše naključen niz. Podpis zahteva potrdilo denarnice Ethereum.

Iz podpisanega niza se potem, spet s pomočjo knjižnice web3.js, izlušči naslov Ethereum, ki je podpisal niz. Če sta izluščen naslov Ethereum in posredovan naslov Ethereum enaka, potem je uporabnik dokazal, da je lastnik tega naslova [43].

Po dokazilu lastništva računa Ethereum je treba preveriti, ali si ta račun lasti plačniško vstopnico. To se preveri preko knjižnice web3.js, ki kliče funkcijo na pametni pogodbi, ki to preveri. V primeru, da je lastništvo potrjeno, je uporabnik dokazal, da ima kupljeno vstopnico.

Plačniške vstopnice niso prenosljive, torej ni potrebe po dodatni logiki, ki bi preprečila večkratno uporabo istih kart.

5.5 Način testiranja

5.5.1 Testiranje pametne pogodbe


```
{
   "blockHash":"0x9ed66afa07db40e50c734c8480ef160d6a1b0f7348144e887d39f3f3889e7e3d",
   "blockNumber":4030781,
   "contractAddress":null,
   "cumulativeGasUsed":2958266,
   "from":"0xc42e6376016f11c63da93a508dc15578eb3c247b",
   "gasUsed":411069,
   "logs":{
      "0":{
         "address":"0xe28671e9d4f33e6bbb8695a4f82b2c90de1334eb",
         "blockHash":"0x9ed66afa07db40e50c734c8480ef160d6a1b0f7348144e887d39f3f3889e7e3d",
         "blockNumber":4030781,
         "data":"0x000000000000000000000000000000000000000000000000000000000000000200000000000...",
         "logIndex":5,
         "removed":false,
         "topics":[
            "0xd7789a3deb1d0944b7966c5d18037fc5b3a0a17236de9e7c20fe3a6345167295"
         ],
         "transactionHash":"0x535050517839d62b3d9f023d0b4803a083bef5797290e24298218ef01c202c67",
         "transactionIndex":5
      },
      "1":"..."
   },
   "logsBloom":"0x00200000000000000000000000000000000000000000000000000000000000000...",
   "status":"0x1",
   "to":"0xe28671e9d4f33e6bbb8695a4f82b2c90de1334eb",
   "transactionHash":"0x535050517839d62b3d9f023d0b4803a083bef5797290e24298218ef01c202c67",
   "transactionIndex":5
}
```


Slika 5.3: Primer izpisa statusa uspešne transakcije Ethereum

Testiranje pametne pogodbe se je izvajalo v razvojnem okolju Remix. Razvojno okolje Remix omogoča, da se za klicanje funkcij pametne pogodbe in preverjanje stanj spremenljivk uporabi kar uporabniški vmesnik. Pri tem

smo si izpisovali stanje transakcij kar prikazuje slika 5.3. Razvojno okolje Remix omogoča preprosto in hitro postavljanje novih pogodb in s tem hitro testiranje novih funkcij ali spremenljivk.

Preverjanje dogajanja znotraj posamezne funkcije se je preverjalo s pomočjo dogodkov (angl. events). Te se je nadzorovalo s pomočjo poslušalcev, ki so bili ustvarjeni s pomočjo knjižnice web3.js. Knjižnica web3.js je služila tudi pri preverjanju stanja vrednosti v pogodbi, s čimer se je poskušalo določiti pravilno količino začetne vrednosti na pogodbi.


```
{
   "asset":{
      "id":"3f5a4bc38b4f8dd52989a2e8860af93627360ef80283cab0072dd28514a43da1"
   },
   "id":"8c8724cfe63009f1d377b23f8d13363cb501ada7656fbe358486d0b23c264d32",
   "inputs":{
      "0":{
         "fulfillment":"pGSAINrc5kkUCqT067_r_Sn2JWPop632SRjeUixq7JkY04i0gUAgBj2xvZHbvIj9D6HzM4ExRi12oe7S2XkIQfIOlMOmOyq56ydep_3z5Uj8NHS-
MUDmZxSB-dEXqXCZLueTlNAM",
         "fulfills":{
            "output_index":"0",
            "transaction_id":"3f5a4bc38b4f8dd52989a2e8860af93627360ef80283cab0072dd28514a43da1"
         },
         "owners_before":{
            "0":"FjMF5mRuNNydfVdY4eADwF4ydZ6pXfnPwz1fVYXjYUuh"
         }
      },
      "metadata":null,
      "operation":"TRANSFER",
      "outputs":{
         "0":{
            "amount":"20",
            "condition":{
               "details":{
                  "public_key":"FjMF5mRuNNydfVdY4eADwF4ydZ6pXfnPwz1fVYXjYUuh",
                  "type":"ed25519-sha-256"
               },
               "uri":"ni:///sha-256;apfji5w78y-e98kWvDi76N0VqzR0SdpgJ5neCTCaAMg?fpt=ed25519-sha-256&cost=131072"
            },
            "public_keys":{
               "0":"FjMF5mRuNNydfVdY4eADwF4ydZ6pXfnPwz1fVYXjYUuh"
            }
         },
         "1":{
            "amount":"5",
            "condition":{
               "details":{
                  "public_key":"5jAsBhVvrwyDrbSJzWEdbn6xSwGr6GS5g4txCH6Pex8K",
                  "type":"ed25519-sha-256"
               },
               "uri":"ni:///sha-256;lEbcJSL8i-KJd2JGdxNBvW2YQHIFmZwJQs-MRy-3ci4?fpt=ed25519-sha-256&cost=131072"
            },
            "public_keys":{
               "0":"5jAsBhVvrwyDrbSJzWEdbn6xSwGr6GS5g4txCH6Pex8K"
            }
         }
      }
   },
   "version":"2.0"
}
```


Slika 5.4: Primer izpisa prenosne transakcije BigchainDB

Delo z decentralizirano podatkovno bazo BigchainDB se je testiralo s pomočjo izpisovanja transakcij, ki so bile sprejete v omrežje BigchainDB. Primer izpisa transakcije je prikazan na sliki 5.4. Do transakcij se lahko

dostopa v brskalniku tudi preko vmesnika HTTP [23] ali s pomočjo gonilnika BigchainDB [14].

V namen testiranja se je ustvarilo tudi več uporabniških računov BigchainDB, ki so si med testiranjam izmenjevali lastništva nad vstopnicami.

Poglavje 6

Zaključek

Glavni cilj diplomske naloge smo realizirali s proučitvijo področja decentraliziranih aplikacij in področja decentraliziranega hranjenja podatkov. Opisali smo koncept decentraliziranih aplikacij in verige blokov, ki je osnova decentralizacije. Natačneje smo pregledali platformo Ethereum in gradnjo decentraliziranih aplikacij na tej njej. Pri decentraliziranem hranjenju podatkov smo se osredotočili na decentralizirano podatkovno bazo BigchainDB in njene koncepte. Proučili smo tudi koncept oraklov, ki so potrebni za komunikacijo med verigo blokov Ethereum in decentralizirano podatkovno bazo BigchainDB.

S pomočjo praktičnega primera smo prikazali današnje omejitve decentraliziranih aplikacij. Glavna težava, ki jo je potrebno rešiti je hitrost sprejemanja transakcij, saj gradnja hitro odzivnih aplikacij trenutno ni mogoča. Omejitev je tudi v težavnosti izvajanja aplikacij, ki jih lahko gosti platforma Ethereum. Cena plina raste hitro s kompleksnostjo pametnih pogodb, zato je razvoj aplikacij precej omejen. Dodatna težava je pridobivanje podatkov izven omrežja Ethereum. Uporaba oraklov dodatno poveča neodzivnost aplikacije in lahko zmanjša zaupanje med uporabniki aplikacije in njenimi razvijalci. Praktičen primer in opis omejitev smo naredili v okviru drugega cilja diplomske naloge, ki je bil s tem uspešno izpolnjen.

Prvoten načrt praktičnega primera je predvidel več komunikacije med de-

centralizirano aplikacijo Ethereum in decentralizirano podatkovno bazo BigchainDB. Ugotovili smo, da to ni mogoče zaradi omejitev pri decentraliziranih aplikacijah. Reševanje teh omejitev ostaja naloga za nadaljni razvoj. Z njihovo odpravo bodo decentralizirane aplikacije postale enakovredna rešitev klasičnim aplikacijam.

Literatura

[1]

[2] Atin Angrish, Benjamin Craver, Mahmud Hasan, and Binil Starly. A case study for blockchain in manufacturing: "fabrec": A prototype for peer-to-peer network of manufacturing nodes. *Procedia Manufacturing*, 26:1180 – 1192, 2018. 46th SME North American Manufacturing Research Conference, NAMRC 46, Texas, USA.

[3] Rupali Arora and Rinkle Rani Aggarwal. Modeling and querying data in mongodb. *International Journal of Scientific and Engineering Research*, 4(7):141–144, 2013.

[4] Imran Bashir. *Mastering Blockchain*. Packt Publishing Ltd, 2017.

[5] Test network documentation. Dosegljivo: `https://testnet.bigchaindb.com/docs/`, 2018. [Dostopano: 3. 9. 2018].

[6] A decentralized database for the future. Dosegljivo: `https://s3.amazonaws.com/bluzelle-craft-private-storage/Bluzelle-Whitepaper-English.pdf`, 2017. [Dostopano: 4. 8. 2018].

[7] Bootstrap. Dosegljivo: `https://getbootstrap.com/`, 2018. [Dostopano: 2. 9. 2018].

[8] Burn/delete. Dosegljivo: `https://tutorials.bigchaindb.com/crab/burn`, 2018. [Dostopano: 2. 9. 2018].

[9] Roberto Casado-Vara, Javier Prieto, Fernando De la Prieta, and Juan M. Corchado. How blockchain improves the supply chain: case study alimentary supply chain. *Procedia Computer Science*, 134:393 – 398, 2018. The 15th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2018) / The 13th International Conference on Future Networks and Communications (FNC-2018) / Affiliated Workshops.

[10] Decentralized oracles: Why ethereum needs chainlink, and not the other way around. Dosegljivo: `https://fynestuff.com/chainlink-vs-decentralized-oracles-analysis/`, 2018. [Dostopano: 12. 8. 2018].

[11] Bitinfocharts. Dosegljivo: `https://blog.fluence.ai/a-decentralized-database-for-decentralized-internet-e28a32b983f0`, 2018. [Dostopano: 13. 8. 2018].

[12] Decentralized apps vs web apps. Dosegljivo: `https://blog.wiredelta.com/decentralized-apps-vs-web-apps/`, 2018. [Dostopano: 8. 8. 2018].

[13] Alexander Demidko, Evgeny Ponomarev, Dmitry Kurinskiy, Constantine Solovev, and Artemiy Liarskiy. Fluence: A decentralized database network. 2018.

[14] drivers. Dosegljivo: `https://docs.bigchaindb.com/projects/server/en/latest/drivers-clients/index.html`, 2018. [Dostopano: 6. 8. 2018].

[15] Ed25519: high-speed high-security signatures. Dosegljivo: `https://ed25519.cr.yp.to/`, 2017. [Dostopano: 5. 8. 2018].

[16] Ethereum. Dosegljivo: `https://www.ethereum.org/`, 2018. [Dostopano: 7. 8. 2018].

[17] Ethereum development tutorial. Dosegljivo: https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial, 2018. [Dostopano: 8. 8. 2018].

[18] A next-generation smart contract and decentralized application platform. Dosegljivo: https://github.com/ethereum/wiki/wiki/White-Paper, 2018. [Dostopano: 8. 8. 2018].

[19] Why many smart contract use cases are simply impossible. Dosegljivo: https://www.coindesk.com/three-smart-contract-misconceptions/, 2016. [Dostopano: 10. 8. 2018].

[20] Ropsten ethereum faucet. Dosegljivo: https://faucet.ropsten.be/, 2018. [Dostopano: 2. 9. 2018].

[21] A decentralized database for decentralized internet. Dosegljivo: https://blog.fluence.ai/a-decentralized-database-for-decentralized-internet-e28a32b983f0, 2018. [Dostopano: 7. 8. 2018].

[22] BigchainDB GmbH. Bigchaindb 2.0 the blockchain database. 2018.

[23] The http client-server api. Dosegljivo: https://docs.bigchaindb.com/projects/server/en/latest/http-client-server-api.html, 2018. [Dostopano: 6. 8. 2018].

[24] jquery. Dosegljivo: https://jquery.com/, 2018. [Dostopano: 2. 9. 2018].

[25] The easiest way to deploy smart contracts on ethereum. Dosegljivo: https://medium.com/@marcusmolchany/the-easiest-way-to-deploy-smart-contracts-on-ethereum-65f69ac9a627, 2018. [Dostopano: 9. 8. 2018].

[26] Mobius network will help any developer become a blockchain developer. Dosegljivo: https://hackernoon.com/mobius-network-will-help-

any-developer-become-a-blockchain-developer-b39fd5f3837f, 2018. [Dostopano: 13. 8. 2018].

[27] Off-chain data storage: Ethereum & ipfs. Dosegljivo: https://medium.com/@didil/off-chain-data-storage-ethereum-ipfs-570e030432cf, 2017. [Dostopano: 7. 9. 2018].

[28] Home. Dosegljivo: https://docs.oraclize.it/, 2018. [Dostopano: 12. 8. 2018].

[29] M Tamer Özsu and Patrick Valduriez. *Principles of distributed database systems*. Springer Science & Business Media, 2011.

[30] Narayan Prusty. *Building Blockchain Projects*. Packt Publishing Ltd, 2017.

[31] Matevž Pustišek and Andrej Kos. Approaches to front-end iot application development for the ethereum blockchain. *Procedia Computer Science*, 129:410 – 419, 2018. 2017 INTERNATIONAL CONFERENCE ON IDENTIFICATION,INFORMATION AND KNOWLEDGEIN THE INTERNET OF THINGS.

[32] Simple http request handler. Dosegljivo: https://docs.python.org/2/library/simplehttpserver.html, 2018. [Dostopano: 3. 9. 2018].

[33] Welcome to remix documentation! Dosegljivo: https://remix.readthedocs.io/en/latest/, 2018. [Dostopano: 2. 9. 2018].

[34] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with iot. challenges and opportunities. *Future Generation Computer Systems*, 88:173 – 190, 2018.

[35] What is tendermint? Dosegljivo: https://tendermint.readthedocs.io/en/master/introduction.html, 2018. [Dostopano: 5. 8. 2018].

[36] Bigchaindb testnet. Dosegljivo: https://testnet.bigchaindb.com/, 2018. [Dostopano: 6. 8. 2018].

[37] Stefan Thomas and R Reginelli. Crypto-conditions. *GitHub. https://github. com/interledger/rfcs/tree/master/0002-cryptoconditions*, 2016.

[38] Bigchaindb transactions spec v2. Dosegljivo: `https://github.com/bigchaindb/BEPs/tree/master/13`, 2018. [Dostopano: 5. 8. 2018].

[39] Transaction concepts. Dosegljivo: `https://docs.bigchaindb.com/en/latest/transaction-concepts.html`, 2018. [Dostopano: 5. 8. 2018].

[40] Transactions speeds: How do cryptocurrencies stack up to visa or paypal? Dosegljivo: `https://howmuch.net/articles/crypto-transaction-speeds-compared`, 2018. [Dostopano: 7. 9. 2018].

[41] The easiest way to deploy smart contracts on ethereum. Dosegljivo: `https://truffleframework.com/docs/truffle/getting-started/truffle-with-metamask`, 2018. [Dostopano: 9. 8. 2018].

[42] Sricharan Vadapalli. *Hands-on DevOps*. Packt Publishing, 2017.

[43] How to verify metamask account holder is the real owner of the address? Dosegljivo: `https://ethereum.stackexchange.com/questions/35486/how-to-verify-metamask-account-holder-is-the-real-owner-of-the-address`, 2018. [Dostopano: 3. 9. 2018].

[44] Žiga Turk and Robert Klinc. Potentials of blockchain technology for construction management. *Procedia Engineering*, 196:638 – 645, 2017. Creative Construction Conference 2017, CCC 2017, 19-22 June 2017, Primosten, Croatia.