Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Simon Oberžan

Tehnologija veriženja blokov in e-volitve

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM
PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: izr. prof. dr. Erik Štrumbelj

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Študent naj v diplomski nalogi predstavi tehnologijo veriženja blokov in kako bi jo lahko uporabili za e-glasovanje. Izvedljivost takega pristopa naj utemelji z izdelavo prototipne aplikacije, ki bo omogočala osnovne funkcije pri volitvah - registracijo uporabnikov, izvedbo glasovanja in povzetek rezultatov.

Ob tej priložnosti bi se rad zahvalil mentorju Eriku Štrumblju za potrpežljivost in pomoč pri pisanju diplomske naloge. Prav tako bi se rad zahvalil za vse skupno sodelovanje in strokovno pomoč v času študija. Velika zahvala gre staršema in sestri za podporo ob študiju ter punci Tini, ki me je prenašala med pisanjem diplomske naloge.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ICO	initial coin offering	začetna ponudba kovancev
CA	certificate authority	izdajatelj pooblastil
MSP	membership service provider	ponudnik članstvenih storitev
HA	high availability	visoka razpoložljivost
KVS	key-value store	shramba tipa ključ-vrednost
DNS	Domain Name Server	strežnik domenskih imen
API	application programming interface	aplikacijski programski vmesnik
SMTP	Simple Mail Transfer Protocol	enostaven protokol za prenos elektronske pošte
SOAP	Simple Object Access Protocol	enostaven protokol za izmenjavo sporočil
CT	crash tolerance	toleranca izpada
POW	proof-of-work	dokazilo o delu
REST	Representational State Transfer	način interoperabilnosti med računalniškimi sistemi
PIN	Personal Identification Number	osebna identifikacijska številka
URL	Uniform Resource Locator	enolični krajevnik vira
SDK	Software Development Kit	paket za razvoj programske opreme
DoS	Denial of Service	zavrnitev storitve

Povzetek

Naslov: Tehnologija veriženja blokov in e-volitve

Avtor: Simon Oberžan

Volitve so eden izmed temeljev, na katerih sloni demokracija. Klasični proces volitev je še vedno oddajanje označenih glasovalnih listkov, ki se jih pred razglasitvijo rezultata ročno prešteje. Ta proces je brez večjih sprememb enak že tisoče let in je eden izmed zadnjih procesov, ki še ni v elektronski obliki. Avtoriteta, ki vodi volitve, ima na ta način večji vpliv pri procesu, hkrati pa je ta tudi bolj ranljiv proti zlonamernim entitetam, ki želijo vplivati na rezultate volitev. Cilj moje diplomske naloge je bil spoznati probleme elektronskega glasovanja ter implementirati rešitev glasovanja s tehnologijo veriženja blokov, ki poseduje mnoge lastnosti, ki nam lahko koristijo pri implementaciji rešitve. Takšna aplikacija mora biti implementirana transparentno in biti preprosta za uporabo, da je glasovanje za volivce enostavnejše ter da mu ti lahko zaupajo.

Ključne besede: tehnologija veriženja blokov, volitve, glasovanje.

Abstract

Title: Blockchain and e-voting

Author: Simon Oberžan

Voting is one of the cornerstones of democracy. The classic election process is still based on casting of marked ballots, which are then counted manually before announcing the results. This process has gone unchanged for thousands of years and is one of the last processes that is not yet in an electronic form. The authority leading such elections has greater influence, and the process is also more vulnerable to malicious entities that want to influence the election results. The goal of my Bachelor's thesis was to get to know the problems of electronic voting and to implement a solution of voting with the blockchain technology, which has many beneficial properties. Such applications must be implemented in a transparent manner and be easy to use so that the process is trustworthy and simple to use for the voter.

Keywords: blockchain, elections, voting.

Poglavje 1

Uvod

1.1 Motivacija

Demokracija je temelj, na katerem sloni večina razvitega sveta. Temelji na vključevanju državljanov v proces upravljanja države in odločanja smeri, v katero jo želimo popeljati. Da pa lahko prepoznamo voljo množice ljudi, prirejamo volitve, kjer ti oddajo svoje glasove. Učinkovitost in poštenost volitev je odvisna od državne organizacije, ki volitve izvaja. Še vedno je namreč najpogostejši način voljenja z označevanjem pripravljenih anonimnih listkov ter kasnejše preštevanje glasov. To dopušča veliko možnosti za zlorabo procesa, ki temelji na tem, da glasovanje poteka fizično. Prav tako je pri takšnem glasovanju potrebna fizična prisotnost volivcev, kar se lahko izkaže za problematično, saj volivci niso sistematično zastopani – nekatere skupine težje najdejo čas, voljo ali si udeležbe preprosto ne morejo privoščiti.

Mnoge izmed teh problemov bi bilo možno rešiti z uporabo elektronskega načina glasovanja. To je sicer bila možnost že vrsto let, ampak je tehnologija, ki je bila na voljo, prinesla preveč pomankljivosti, ki so onemogočala izvedbo zaupanja vrednih volitev. Dandanes pa se vedno bolj uveljavlja tehnologija veriženja blokov, ki omogoča rešitve, ki prej niso bile možne. Ker sta mi tematiki demokracije in tehnologije zelo pri srcu, sem se odločil, da raziščem to področje ter prispevam k hitrejšemu napredku.

1.2 Cilj

Cilj te naloge je raziskati tehnologije Hyperledger in z njihovo uporabo implementirati elektronsko glasovanje. Hyperledger Fabric je odprtokodno ogrodje pod okriljem Fundacije Linux, ki omogoča postavitev omrežja verig blokov ter pisanje in izvajanje pametnih pogodb. V prvem delu naloge bomo vzpostavili vse potrebne komponente za delovanje omrežja. Kljub temu, da je omrežje možno enostavno postaviti na enem računalniku, ga bomo postavili v HA-načinu delovanja, ki je potreben za produkcijske namene. V drugem delu pa bomo spisali pametne pogodbe, ki bodo definirale potek volitev, in implementirali grafični vmesnik, s katerim bodo udeleženci lahko enostavno ustvarili nove volitve, oddali glas oziroma pregledali rezultate volitev.

1.3 Zgradba

V diplomskem delu najprej opišemo tehnologijo veriženja blokov, njene lastnosti, kako je nastala in pregledamo trenutno stanje, pri čemer naredimo primerjavo z drugimi produkti, ki so nam na voljo. V tretjem poglavju opišemo odprtokodni projekt Hyperledger, katerega ogrodje Fabric smo uporabili za implementacijo glasovalne aplikacije. Poleg ogrodja Fabric opišemo še orodji Hyperledger Cello in Hyperledger Composer, saj smo ju uporabili pri postavitvi omrežja verig blokov in definiranju poslovnega omrežja.

Sledi še pregled področja elektronskega glasovanja, opišemo prednosti in pomanjkljivosti, njegove začetke ter opišemo trenutno stanje.

V zadnjem poglavju opišemo našo aplikacijo. To poglavje je razdeljeno na tri sklope. V prvem se poglobimo v postopek postavitve omrežja verig blokov, kjer smo postopek avtomatizirali z močno prilagoditvijo že obstoječih skript pod projektom Hyperledger Cello. V drugem poglavju opišemo, kako je z orodjem Hyperledger Composer definirano naše poslovno omrežje. V tretjem sklopu opišemo še našo spletno aplikacijo, ki služi kot uporabniški vmesnik za interakcijo z našim poslovnim omrežjem.

Poglavje 2

Tehnologija veriženja blokov

Leta 2008 je Satoshi Nakamoto izdal dokument, v katerem je opisal nov pristop k implementaciji elektronskega denarja, ki ga je poimenoval *Bitcoin* [11]. Za njegovo implementacijo je opisal novo tehnologijo, ki temelji na veriženju blokov, kasneje poimenovano „blockchain“ – veriga blokov. Ta tehnologija močno temelji na kriptografičnih metodologijah, kot so simetrični ključi in zgoščevalne (ang. hashing) funkcije.

Po uspešni implementaciji omrežja Bitcoin, je potencial tehnologije kmalu postal jasen. Jedro tehnologije veriženja blokov je odprta, porazdeljena glavna knjiga, ki omogoča izvajanje in shranjevanje transakcij med več udeleženci na permanenten način brez potrebne zunanje avtentikacije. Tehnologijo lahko torej uporabljamo za mnogo različnih aplikacij, ne samo za implementacijo valut oziroma kriptovalut, kot je Bitcoin.

Leta 2013 je Vitalik Buterin, soustanovitelj revije Bitcoin Magazine, predlagal razširitev tehnologije Bitcoin z implementacijo zmogljivejšega skriptnega jezika. Ker predlog ni uspel pridobiti zanimanja v skupnosti Bitcoin, je objavil in kasneje razvil novo tehnologijo Ethereum, katere omrežje je postalo javno leta 2014. Ethereum je uporabnikom omogočal definiranje lastnih pametnih pogodb, kot so poimenovane aplikacije, ki se nato izvajajo v omrežju v zameno za plačilo v lastni kriptovaluti Ether [13]. To je odprlo vrata novim aplikacijam, ki želijo lastnosti omrežja blokov prenesti na procese vseh vrst

industrij.

Ethereum je vzbudil veliko zanimanje med razvijalci, ki so začeli implementirati nove vrste aplikacij. K popularnosti so pripomogle tudi velike količine denarja, zbrane z ICOti, kot se je poimenovala začetna prodaja deleža žetonov, ustvarjenih na omrežju verig blokov. Čeprav je Ethereum precej hitrejši od omrežja Bitcoin, pa je tudi ta kmalu naletel na težave preobremenjenosti. To je povzročilo počasnejše in dražje izvajanje transakcij, kar je otežilo implementacijo aplikacij. Za razliko od Bitcoina pa ima Ethereum aktivno omrežje razvijalcev in veliko podporo iz industrije, kar omogoča razvijanje izboljšav, ki bodo pripomogle k odpravi njegovih pomanjkljivosti.

Kljub uporabnosti tehnologije blokov se ta ni močno prijela med velikimi organizacijami. Pomanjkljivosti, kot so cena transakcij, nizka hitrost omrežja in anonimnost uporabnikov, namreč zanje pogosto predstavljajo problem. Podjetja za razliko od končnih uporabnikov pogosto ne želijo anonimnosti v omrežjih, saj to oteži sledljivost in morda celo nasprotuje internim pravilnikom. Za potrebe takšnih organizacij je bil implementiran projekt Hyperledger, ki omogoča postavitev zasebnih omrežij s pooblaščenimi udeleženci, ki so lahko za organizacije precej cenejša in ponujajo velik napredek na področju hitrosti izvajanja transakcij.

2.1 Osnovno delovanje

Omrežje Blockchain sestavljajo vozlišča, ki hranijo in posodabljajo glavno knjigo. Različne tehnologije imajo različne pristope, ampak večinoma ta vozlišča sestavljajo objavljene transakcije v bloke, ki jih nizajo v glavno knjigo. Ti bloki med drugim vsebujejo tudi zgoščeno vrednost prejšnjega bloka [11]. Ker zgoščevalna funkcija kot vhodne parametre prejme vse atribute bloka, je med drugim odvisna tudi od zgoščene vrednosti prejšnjega bloka. Tako kakršnakoli sprememba na kateremkoli bloku v glavni knjigi spremeni zgoščeno vrednost naslednjega bloka, ki nato spremeni zgoščeno vrednost naslednjega bloka itd. Če želi vozlišče v omrežjih brez potrebnega

dovoljenja (ang. permissionless) ustvariti blok, sprejet po celotnem omrežju, mora biti njegova zgoščena vrednost manjša od trenutno zahtevanega števila, ki se ciklično posodablja za ohranjanje konstantnega povprečnega časa med bloki. Vozlišča takšne bloke iščejo tako, da v blok vstavijo dodatno število, imenovano nonce. Če zgoščena vrednost bloka ne ustreza pogojem, vozlišče preprosto poveča nonce in poskusi ponovno [6]. Vozlišča nato poskušajo čim hitreje najti primeren blok in se ustavijo le, ko jim to uspe ali ko prejmejo ustrezen blok iz omrežja. Vozlišča tu med sabo tekmujejo in poskušajo čim prej najti primeren blok ter ga nato razposlati po omrežju. Ko ostala vozlišča prejmejo blok, preverijo njegovo ustreznost, in če določijo, da je pravilen, običajno začnejo graditi blok, ki ga bo nasledil. Vozlišča torej običajno gradijo na najdaljši znani verigi, saj je ta z njihovega stališča verjetno pravilna, saj predvidevajo, da večina udeležencev ne poskuša goljufati. Tu seveda obstaja možnost, da želi zlonamerno vozlišče spremeniti verigo, a bi za to potrebovalo večino vseprocesorske moči, da bi lahko bloke ustvarjalo hitreje od vseh drugih udeležencev v omrežju [11].

2.2 Lastnosti

Verige blokov z vključevanjem zgoščenih vrednosti bloka v naslednji blok, dosežejo enostavno dokazovanje nespremenljivosti. Če pride do spremembe v bloku, je ta takoj očitna, saj se morajo za oblikovanje veljavne verige poleg zgoščene vrednosti trenutnega bloka spremeniti tudi zgoščene vrednosti vseh nadaljnjih blokov. Vozlišča sledijo in gradijo na najdaljši znani verigi blokov, saj je delo na verigi, ki ni sprejeta na celotnem omrežju, nenagrajeno. Ti dejstvi torej pomenita, da bi vozlišče, ki bi želelo vsiliti spremembo blokov, moralo ustrezne bloke ustvarjati hitreje kot celotno preostalo omrežje skupaj [11]. V primeru verig blokov z doseganjem soglasja z dokazovanjem dela bi to pomenilo, da zlonamerno vozlišče oziroma omrežje potrebuje več kot 50 % procesorske moči. V verigah blokov z doseganjem soglasja z zastavljanjem valut pa bi to pomenilo, da ta potrebuje več kot 50 % vseh valut. Takšna

vrsta napada je izjemno težko izvedljiva s strani kateregakoli udeleženca, a je tu problem, saj se rudarji velikokrat povezujejo v rudarska združenja (ang. mining pools), katerim oddajajo svojo procesorsko moč, v zameno pa prejmejo sorazmerni delež nagrade, ki jo ta prejme. Problem nastane, ker takšna združenja velikokrat zberejo veliko količino procesorske moči, kar njihovim administratorjem omogoča zelo veliko moč, na primer napad dvojnega zapravljanja (ang. double spend) [6]. Pred takšnimi napadi je obramba tudi dejstvo, da ima omrežje z večino procesorske moči oziroma zastavljene valute tudi največji vložek in bi ob napadu bilo močno oškodovano.

Zanimiva lastnost tehnologije verig blokov je tudi omogočanje anonimnosti. Ta se je izkazala kot izjemno zanimiva predvsem pri implementaciji valut, saj ponuja alternativo storitvam klasičnega finančnega sistema, ki uporabnike identificira na vsakem koraku. Uporabniki namreč za kreiranje računa potrebujejo le par ključev, pri čemer javni ključ predstavlja njihovo identiteto, z zasebnim ključem pa uporabniki dokažejo lastništvo računa in avtorizirajo svoje transakcije. Obljubljena anonimnost je lahko tudi past za uporabnike, ki ne poznajo tehnologije, saj gre le za psevdoanonimnost [9]. Uporabnik sicer res lahko ustvari nov račun, s katerim izvaja transakcije, in s tem ne izda svoje identitete, a je ta lahko hitro ogrožena, saj je uporabnikovo identiteto mogoče izpeljati iz njegovih transakcij. Na primer, če izvedemo analizo uporabniških transakcij z računi, katerih lastniki so identificirani (s svojimi drugimi računi, računi prijateljev ali lastnim računom na menjalnicah, kjer je potrebna identifikacija), lahko z veliko natančnostjo predvidevamo lastnika računa. Ko je njihova identiteta povezana z njihovim računom, uporabniki izgubijo svojo anonimnost, saj je zaradi javnosti omrežja zdaj mogoče slediti vsem njihovim transakcijam. Ta problem povezovanja računov je bil izpostavljen že v izvirnem dokumentu Bitcoin [11].

Ker je večina omrežij verig blokov javnih in ne moremo predvidevati, da bodo vozlišča delovala v skladu s pravili, morajo ta omogočati odpornost na bizantinske napake – BFT. To pomeni, da mora naše omrežje ohraniti pravilno delovanje, tudi če vemo, da bodo določeni udeleženci v omrežje aktivno

podajali napačne informacije. Vsa vozlišča ob prejemu blokov preverijo njihovo veljavnost, nato pa ga sprejmejo le v primeru, da je ta del njim najdaljše znane verige [11], kar pomeni, da je bil ustvarjen s strani omrežja z večino moči, bodisi procesorske ali pa poseduje večino valute. Takšne odpornosti ponavadi ne najdemo v klasičnih sistemih, na primer v omrežjih bank, pri katerih se transakcije izvajajo le na lastnih strežnikih v zavarovanem omrežju in se predvideva, da vsa vozlišča delujejo po pravilih. Ta lastnost omogoča, da lahko sisteme, ki so zgodovinsko gledano bili implementirani in vodeni s strani avtoritet z veliko močjo nad celotnim procesom, implementiramo porazdeljeno in tako, da nobena entiteta ne mora uvesti sprememb brez privoljenja večine udeležencev omrežja.

2.3 Vrste tehnologij

Bitcoin je prva aplikacija, narejena z uporabo tehnologije verige blokov. Valute so namreč ena najprimernejših aplikacij za tehnologijo, saj ponujajo enostaven način spodbujanja vozlišč k njihovi udeležbi v omrežju. Bitcoin uporabnikom omogoča anonimno ustvarjanje računov in opravljanje transakcij. Prav tako zagotovlja, da avtoritete, kot so banke ali države, uporabniških računov ne morejo zamrzniti ali blokirati. Celotno delovanje omrežja je povsem transparentno in je obljubljalo poceni izvajanje transakcij. To je bil velik nabor prednosti pred klasičnimi finančnimi sistemi, in to ravno v času, ko je bilo zaupanje v banke rekordno nizko [8]. A Bitcoin je zapolnil le nišo v finančnem sistemu, saj se je veliko lastnosti izkazalo za vprašljivih, na primer anonimnost, ki je v mnogih primerih zavajujoča, ali pa cena transakcij, ki se je močno dvignila. Zaradi teh namenov, kot tudi zaradi potreb po implementaciji aplikacij poleg valut je nastalo mnogo novih tehnologij, ki so si različne v nekaterih bistvenih konceptih. V tem poglavju so tehnologije verig blokov razdeljene na večje skupine različnih vrst implementacij.

Velika skupina tehnologij, ki se močno razlikuje od Bitcoina kot največjega predstavnika valute, omogoča izvajanje pametnih pogodb. Prvi in največji

predstavnik te skupine je Ethereum, ki je tehnologijo verig blokov popeljal v precej drugo smer kot enonamenski Bitcoin, ki primarno deluje kot valuta. Čeprav je bilo verigo blokov Bitcoin mogoče uporabiti tudi za druge aplikacije, na primer za hrambo zgoščenih vrednosti dokumentov, so bile njegove funkcionalnosti za večino razvijalcev prešibke. Uporabniki lahko na omrežju Ethereum implementirajo pametne pogodbe, definirane v določenem skriptem jeziku, in niso več prisiljeni podatkov skrivati v transakcijah, kot so to počeli na omrežju Bitcoin. Takšne pametne pogodbe so lahko precej zahtevnejše od preprostih transakcij na omrežju Bitcoin, zato morajo uporabniki za vsako operacijo pametnih pogodb podati plačilo v obliki lastne valute Ether, ki gre k rudarjem. Takšne tehnologije uporabnikom ne ponujajo le določene aplikacije, temveč platformo, na kateri lahko ti gradijo mnogo vrst lastnih aplikacij. Ker so te tehnologije močno razširile možnosti uporabe tehnologije verig blokov, se zanje pogosto uporablja tudi izraz „blockchain 2.0“ oziroma verige blokov druge generacije [12].

Z rastjo omrežij tehnologij verig blokov, ki je bilo močno spodbujeno z donosnostjo rudarjenja, sta se kot velik problem izkazala poraba električne energije in velika potreba po specializiranih procesorskih enotah. Razlog za to je iskanje soglasja z izračunavanjem primerne zgoščene vrednosti bloka za dokazovanje vloženega dela (PoW), ki je razmeroma enostaven, a neučinkovit pristop. Poleg velike porabe sredstev ima takšen algoritem tudi omejitve glede minimalnega časa, potrebnega med ustvarjanjem novih blokov, kar upočasnjuje hitrost izvajanja transakcij. Prav tako se z veliko porabo povečajo tudi stroški rudarjev, ki pa se posledično prenesejo na uporabnike preko zaračunanih stroškov za izvajanje transakcij. Zaradi teh razlogov je mnogo tehnologij implementiralo alternativne pristope, ki imajo svoje prednosti in slabosti. Poleg PoW je verjetno najpopularnejši pristop iskanja soglasja z dokazovanjem vložka (PoS). Temu algoritmu, ki je bil prvotno zasnovan in implementiran na omrežju PeerCoin, se je priljubljenost povečala, saj drugo največje javno omrežje verig blokov Ethereum aktivno razvija pristop doseganja soglasja s PoS, ki bo v prihodnosti zamenjal PoW. Osnovno

načelo tega algoritma je doseganje porazdelejenega soglasja na način glasovanja, pri čemer so glasovi uteženi z relativno vrednostjo valut, ki jih vozlišča posedujejo. Algoritem v primerjavi s PoW opravi enako delo hitreje in z zanemarljivo količino dela. A implementacija v omrežju Ethereum zaradi težav traja dlje, kot je bilo načrtovano. Zaradi enostavnosti glasovanja za blok je vozliščem v interesu glasovati za vse enako dolge verige, saj si tako zagotovijo udeležbo v pravilni verigi in posledično nagrado. PeerCoin je sicer implementiral PoS, a je omrežje dovzetno za to težavo [15]. Ethereum prepozna dve rešitvi problema. Prvi pristop, pogosto poimenovan „slasher“, vozlišča ne samo nagrajuje, temveč tudi kaznuje v primeru, ko ta gradijo bloke na več verigah hkrati [35]. Pri drugem pristopu pa se vozlišča kaznuje, ko ta gradijo na napačni verigi, s čimer je pristop podoben PoW, kjer so rudarji kaznovani tako, da ne dobijo povračila za investicijo elektrike in strojne opreme. Na ta način se rešuje problem takšnega pristopa, saj bi vozlišča sicer lahko glasovala za vse verige hkrati in se s tem izognila glasovanju za napačno verigo [15].

Tehnologije verig blokov se glede na delovanje delijo tudi na takšne, pri katerih za sodelovanje v omrežju ne potrebujemo dovoljenja, na primer Bitcoin ali Ethereum, in takšne, pri katerih moramo predhodno pridobiti identiteto in avtorizacijo, na primer Hyperledger Fabric. Tehnologije brez dovoljenja so javna omrežja, v katerih lahko sodeluje vsakdo s kreiranjem transakcij ali kot vozlišče, ki jih pomaga izvajati. Pri tem sodelujočim ni treba izdati svoje osebnosti ali pridobiti kakršnegakoli dovoljenja. Račun, ki je ponavadi predstavljen kot par ključev, lahko namreč ustvarijo kar brez povezave. Pri tehnologijah z dovoljenjem pa so vsi sodelujoči znani. Takšna omrežja so manjša, delujejo le znotraj točno določenih organizacij, ki imajo skupno omrežje za reševanje skupnih oziroma povezanih problemov. Vsi akterji imajo poleg identitete tudi določeno stopnjo avtorizacije, ki jim je lahko tudi odvzeta. Dobra stran takšnih omrežij je, da lahko večini akterjev v omrežju zaupamo, kar nam omogoča implementacijo hitrejših in razširljivejših algoritmov za doseganje soglasja. Prav tako je zaradi znanosti in manjšega števila udeležencev

lažje pridobiti soglasje, ko želimo uvesti spremembo glede delovanja samega omrežja.

2.4 Trenutno stanje

Bitcoin je danes še vedno največje omrežje po številu uporabnikov, procesorski moči, skupni vrednosti vseh kovancev in prepoznavnosti. To izhaja iz njegove preprostosti in zaupanja v sistem zaradi dolge zgodovine delovanja, prve implementacije tehnologije in zgodb o izjemnih zaslužkih z njegovim trgovanjem. Obstaja pa mnogo novih tehnologij, ki so gradile na njegovi osnovi. Ena izmed najpopularnejših novih skupin tehnologij, katere glavni predstavnik je Ethereum, omogoča porazdeljeno izvajanje aplikacij, ponavadi imenovanih pametne pogodbe. Kot Bitcoin tudi Ethereum za pridobivanje soglasja uporablja pristop dokazila o delu, a že ima načrte za uporabo kompleksnejšega pristopa z dokazilom o vložku. Tako omrežjem za razliko od rudarjenja pri tem konceptu ne bo treba izvajati ogromnega števila operacij, kar bo bistveno zmanjšalo porabo velike količine električne energije in potrebne strojne opreme. Ethereum kot tudi večina drugih tehnologij verig blokov pa se spopada tudi s težavo, da omrežje preprosto ni dovolj razširljivo. Glavni razlog za to je, da morajo vsi rudarji v omrežju izvesti in potrditi vse transakcije. V prihodnosti se želi Ethereum tega problema rešiti z implementacijo algoritma, ki so ga poimenovali „sharding“ in bo omogočal, da bo moral transakcije izvesti le del omrežja. S tem bi naredili velik korak naprej, saj bi to končno pomenilo, da s povečevanjem števila rudarjev povečamo zmogljivosti celotnega omrežja.

Z implementacijo Ethereum se je močno povečalo zanimanje razvijalcev za tehnologijo veriženja blokov. Zdaj so ti končno dobili priložnost enostavnega definiranja aplikacij in njihovega izvajanja za manjše plačilo. To je privedlo do razvoja velikega števila aplikacij, ki so in še obljubljajo revolucijo v mnogih industrijah, predvsem z implementiranjem izvajanja procesov na takšnem omrežju, s čimer se lahko izognemo posrednikom. Najpopularnejša

vrsta pametnih pogodb omogoča izdajanje novih kriptovalut na omrežju. Te se ustvari s preprosto pametno pogodbo in se z njimi opravlja transakcije podobno kot z valuto Ether. Njihova priljubljenost se je izjemno povečala, saj so predstavljale poslovni model (ICO), ki so ga zagonska podjetja izkoriščala za zbiranje začetnega kapitala. Ethereum je tako našel svojo nišo predvsem med zagonskimi podjetji

Ethereum ni primeren za vse vrste aplikacij. Na časovnico za prihajajoče izboljšave se ne moremo zanašati in določene pomanjkljivosti so za nekatere primere uporabe nesprejemljive. Ker se aplikacije v primerjavi s tradicionalnimi rešitvami med drugim izkažejo za izredno drage, počasne in ne omogočajo zasebne shrambe, se v mnogih omrežjih uporabljajo tehnologije pod okriljem odprtokodnega projekta Hyperledger. Za mnogo podjetij je zasebno omrežje z določenimi in avtoriziranimi uporabniki veliko bolj primerno za njihove potrebe, prav tako pa ta omogoča mnogo hitrejše in zmogljivejše omrežje. Cena tega je, da je takšno omrežje treba postaviti in upravljati, kar močno poveča kompleksnost in količino dela v primerjavi z javnimi omrežji, pri katerih se razvijalci ukvarjajo z aplikacijo, ki se nato izvaja na javnem omrežju. Za večje organizacije oziroma podjetja je to velikokrat cena, ki so jo pripravljeni plačati, kar pojasni njihovo zanimanje za projekt. Med večjimi podporniki projekta so namreč velika podjetja kot na primer IBM, Intel, Cisco, Samsung in Bosch. [16].

Poglavje 3

Hyperledger

Projekt Hyperledger je krovni projekt odprtokodnih tehnologij veriženja blokov in povezanih orodij [17], ki je bil ustanovljen decembra 2015 pod okriljem Fundacije Linux [34]. Razlog za več tehnologij pod okriljem projekta je spodbujanje sodelovanja s strani javnosti za določitev smeri, ki bi tehnologije Hyperledger naredile industrijski standard. Hyperledger Fabric trenutno obsega pet različnih tehnologij veriženja blokov in pet komplementarnih orodij. Izmed tehnologij so trenutno za produkcijo pripravljene le Hyperledger Fabric in Hyperledger Sawtooth, ki sta ogrodji za postavitev omrežja verig blokov za izvajanje pametnih pogodb, in Hyperledger Indy, ki pa je ogrodje za postavitev ekosistema samostojnih identitet na porazdeljeni glavni knjigi [17].

3.1 Hyperledger Fabric

Hyperledger Fabric je ena izmed implementacij tehnologije veriženja blokov pod okriljem projekta Hyperledger, ki nam omogoča izvajanje aplikacij na porazdeljenem omrežju. Namenjena je predvsem uporabi med organizacijami oziroma podjetji, ki pa pogosto prispevajo vire za razvoj tehnologije. Hyperledger Fabric je zasnovan kot modularna rešitev, kar omogoča več inovacij, raznolikosti in boljšo optimizacijo [18]. Omrežje tako lahko sestavljajo

različni tipi posameznih komponent, kot so na primer tip soglasja o potrjevanju blokov, MSP, CA itd. Te lahko določimo glede na naše poslovne zahteve, na primer ali je dovolj, da je naše omrežje odporno na izpade ali mora biti BFT. Ker je poleg zastavljene modularnosti projekt tudi odprtokoden, lahko po potrebi implementiramo oziroma uvozimo neuradne module.

Tehnologija Fabric uporablja pristop, da je za sodelovanje v omrežju potrebno dovoljenje, kar pomeni, da so vsi udeleženci v omrežju znani in potrebno avtorizirani s strani ponudnika članskih storitev (MSP). Ta pogoj je omogočil implementacijo veliko hitrejšega algoritma za doseganje soglasja v omrežju, kjer ne potrebujemo dokazovanja dela in nimamo nikakršnih valut. Hitrost in pretok sta seveda odvisna od podanega omrežja ter aplikacije, ki jo uporabljamo, a testi so pokazali, da je v preprostem primeru implementacije valute v omrežju mogoče izvesti več kot 2000 transakcij na sekundo s povprečno izvedbo transakcije v manj kot pol sekunde [2]. K takšni razširljivosti pripomore tudi dejstvo, da omrežje Hyperledger Fabric ni sestavljeno iz ene vrste vozlišč, kar je v nasprotju z večino javnih tehnologij veriženja blokov. Za izboljšanje razširljivosti je namreč delovanje razdeljeno na vrstnike (ang. peers) in urejevalnike (ang. orderers), katerih naloge so podrobneje opisane v nadaljevanju podpoglavja.

Hyperledger Fabric izstopa kot prvi projekt, ki je omogočal izvajanje aplikacij na porazdeljenem okolju, napisanih v splošnih programskih jezikih (trenutno Go, JavaScript, Java). Drugi projekti so namreč implementirali lastne programske jezike za definiranje delovanja pametnih pogodb, kar pa precej oteži delo razvijalcev. Prav tako je to prvi projekt, ki je omogočal izvajanje takšnih aplikacij brez potrebnega plačevanja za izvajanje v obliki lastne kriptovalute [2]. To je mogoče, saj so vsi uporabniki v omrežju identificirani in avtorizirani.

3.1.1 Vrste vozlišč

V tem podpoglavju opišemo tri glavna vozlišča, ki sestavljajo omrežje Hyperledger Fabric. Za lažjo uporabo so vsa vozlišča podana v obliki vsebnikov

Docker, saj poenostavi delo z vozlišči, ki jih lahko enostavno postavljamo, ustavljamo, spreminjamo, skrbimo za njihovo povezljivost, avtomatizirano postavimo itd.

Vrstnik

Vrstnik je primarno vozlišče v omrežjih Hyperledger Fabric. Vsak vrstnik hrani verižne kode in ima lastno kopijo glavne knjige, v kateri so shranjeni rezultati transakcij. Glavna knjiga je v primeru uporabe LevelDB shranjena kar v vsebniku vrstnika, ob uporabi CouchDB pa je ta shranjena v samostojnem vrstniku. Zaradi hrambe teh podatkov se aplikacije primarno povezujejo na vrstnike za dostop do stanja ali pa klicanje transakcij, za kar uporabljajo Hyperledger Fabric SDK [19], ki je trenutno implementiran za NodeJS in Java [20].

Za komunikacijo z drugimi vozlišči v omrežju se lahko vrstniki včlanijo v kanale. Ti delujejo kot zasebna podomrežja za komunikacijo med vsaj dvema članoma omrežja, kjer lahko vsi udeleženci izvajajo zaupne in zasebne transakcije. Ob priključitvi vrstnika v kanal lahko ta dostopa do skupne glavne knjige [21]. Ker je vrstnik lahko član več različnih kanalov, v katerih so različni člani, se transakcije shranjujejo v posamezne glavne knjige, in to za vsak kanal posebej. To pomeni, da vsi vrstniki hranijo glavno knjigo za vsak kanal, kateremu pripadajo [19]. Po kanalu vrstniki prejmejo enako kopijo novokreiranih blokov, ki jih objavijo urejevalniki. Ti ne potrebujejo povezave do vseh vrstnikov v kanalu, saj se za komunikacijo uporablja opravljalni (ang. gossip) protokol, kjer si podatke o stanju med sabo izmenjujejo tudi vrstniki [22].

Urejevalnik

Urejevalniki skupaj sestavljajo tako imenovano urejevalniško storitev (ang. ordering service). Urejevalniki ne poznajo stanja omrežja in ne sodelujejo le v urejevalni fazi transakcije, kar omogoča modularnost in poenostavi zamenjavo protokola. Torej, po zaključeni izvajalni fazi poteka transakcije od-

jemalec prejme odgovore odobritvenih vrstnikov (ang. endorsement peers). Ko glede na politiko odobritve (ang. endorsement policy) zbere dovolj glasov, jih sestavi v transakcijo in pošlje urejevalniški storitvi za ureditev. [2]. Urejevalniška storitev prejema predloge transakcij in jih ureja v bloke, pri čemer vsaka transakcija vsebuje posodobitev trenutnega stanja in odvisnosti, izračunane v izvajalnem delu transakcije skupaj s kriptografičnimi podpisi potrditvenih vrstnikov. Urejevalniška storitev ves čas prejema transakcijske predloge in jih ureja v bloke z omejeno velikostjo, nato pa jih periodično razpošilja preko kanala do vseh njegovih članov. Ko je blok razposlan, se zaključi urejevalni del transakcije, ki mu na vrstnikih sledi potrjevalni del [19].

Trenutno sta uradno implementirana dva tipa urejevalnih storitev: samostojni urejevalnik – *solo*, ki je namenjen izključno za testiranju, in *kafka*, ki implementira atomično oddajanje sporočil in dopušča izpad dela urejevalnikov. V razvoju je tudi protokol BFT [23], alternativno pa je dopuščena možnost implementacije lastnega protokola za doseganje soglasja.

Certifikatna agencija

Ker je Hyperledger Fabric omrežje s potrebnim dovoljenjem, ima vsak udeleženec lastno identiteto v obliki certifikata X.509, glede na katerega omrežje določi, kdo ima pravice za izvajanje željenih transakcij ali dostop do virov [24]. Za pridobitev identitet za vse udeležence nam Hyperledger Fabric ponuja Docker vsebnik za postavitev certifikatne agencije (oznaka CA), ki upravlja s certifikati. Ta z lastnim certifikatom podpisuje vse nadaljnje certifikate, zato mu morajo zaupati vsi udeleženci v omrežju. To je lahko korenski certifikat, ki poseduje to lastnost [25]. V večjih omrežjih ne uporabljamo izključno korenskega CA za izdajanje identitet, ampak postavimo vmesne CA-je, katerih certifikati so izdani s strani korenskega CA ali drugih vmesnih CA-jev. Podpisovanje certifikatov s strani takšnih CA-jev nam omogoča postaviti verigo zaupanja do vsakega certifikata, izdanega s strani CA-jev v našem omrežju, saj lahko izviru certifikata vedno sledimo do korenskega certifikata.

Zaradi tega moramo biti izredno pazljivi glede varnosti našega korenskega certifikata, saj lahko njegova izraba ogroža varnost celotnega omrežja. Enako seveda velja tudi za vmesne CA-je, a tu ogrozimo le del celotnega omrežja, in sicer dejanja vseh udeležencev, katerih certifikati so bili izdani z njegove strani [24].

Podani vmesnik CA je le privzeta implementacija MSP, ki za delovanje ni nujno potreben. Zaradi modularnosti tehnologije Hyperledger Fabric bi jo lahko tudi zamenjali s katerokoli infrastrukturo zasebnih ključev X.509, ki lahko izdaja vpisne (ang. enrollment) certifikate [4].

3.1.2 Glavna knjiga

Glavna knjiga je nespremenljiva shramba za beleženje transakcij, ki se hrani na porazdeljenem omrežju, kjer si vozlišča ne zaupajo popolnoma. Vsak vrstnik hrani lastno kopijo glavne knjige, do katere lahko neposredno dostopa, za spreminjanje oziroma dodajanje zapisov pa mora poleg izvedbe transakcije prejeti tudi vse potrebne odobritve in na omrežju doseči soglasje o njeni ureditvi [2].

Pri tehnologiji Hyperledger Fabric je glavna knjiga skupni izraz za dve komponenti: verigo blokov in trenutno stanje. Veriga blokov se začne z geneznim blokom, na katerega se nizajo novi bloki, ki vsebujejo transakcije. Za poizvedovanje trenutne vrednosti v glavni knjigi gremo lahko po celotni verigi in izpeljemo trenutno vrednost, a se pri dolgi verigi blokov to izkaže za zelo zamudno. Zaradi tega vrstniki poleg glavne knjige hranijo tudi trenutno stanje, ki pa je shranjeno v podatkovni bazi KVS ter nam omogoča hitro in učinkovito poizvedovanje. Privzeto vrstniki za hrambo stanja uporabljajo LevelDB, ki je vključen v vrstnikov vsebnik, lahko pa uporabljajo Apache CouchDB, ki je ponujen kot samostojni vsebnik in se postavi za vsakega vrstnika posebej. Ta nam omogoča uporabo bogatih in kompleksnih poizvedb in hranjenje stanja v JSON obliki [26]. Z uporabo shrambe CouchDB torej povečamo učinkovitost poizvedb, kadar naše potrebe zahtevajo, da poizvedujemo po atributih, in ne po ključih vrednosti v našem stanju. V praksi se

torej poizveduje po trenutnem stanju, veriga blokov pa služi predvsem kot zgodovinski dnevnik vseh transakcij. Ko vrstniki potrdijo novoprejeti blok, se ta pripne na konec verige blokov, prav tako pa se v skladu z njegovimi izvedenimi transakcijami posodobi trenutno stanje, tako da so vrednosti iz obeh virov enake [2].

3.1.3 Potek izvajanja transakcij

Hyperledger Fabric za izvajanje in potrjevanje transakcij uporablja tok izvedi-uredi-potrdi, za razliko od uredi-izvedi, ki ga uporablja večina tehnologij, od javnih omrežij, kot je npr. Ethereum, do omrežij s potrebnim dovoljenjem, kot so npr. Tendermint, Chain ali Quorum. V takšnem toku se transakcije najprej izvedejo, nato uredijo v bloke z algoritmom za doseganje soglasja nazadnje pa jih preverijo vrstniki, preden jih dodajo na konec verige blokov. Tako se izognemo pomanjkljivostim, kot so zaporedno izvajanje, nedeterministična koda in nezaupnost izvedbe, ki je posledica izvajanja vseh transakcij na vseh vozliščih. Zaporedna izvedba je problem, saj morajo vsi vrstniki zaporedno izvesti vse transakcije, kar močno omejuje pretok omrežja [27]. Prav tako je takšno omrežje ranljivo na napade DoS, saj bi lahko aplikacije z neskončno zanko onesposobile celotno omrežje. Ta težava je v primeru Ethereuma rešena z zaračunavanjem pristojbin za izvajanje pametnih pogodb [13], a to ni uporabno na omrežju Fabric, kjer ne poznamo valut. Problem nedeterministične izvedbe v modelu uredi-izvedi se pojavlja, ker se v zadnjem koraku toka izvedejo transakcije na vseh vrstnikih hkrati, kar lahko privede do razlik v stanjih. Ta problem večina drugih tehnologij rešuje z implementacijo lastnih nedeterminističnih programskih jezikov, ki pa otežujejo delo razvijalcem, ki so se primorani naučiti nov jezik [27].

Izogibanje takšnim pomanjkljivostim poveča uporabnost tehnologije. To se izkaže za posebej kritično pri izvajanju poslovnih transakcij, ki morajo biti izvršene hitro in varno. Transakcijski tok v omrežju Fabric sproži odjemalec z oddajo predloga transakcije vsem odobritvenim vrstnikom izvajane verižne kode, ki so opisani v politiki odobravanja. Ta vsebuje identifikator verižne

kode in njene parametre, vsebino transakcije in njen identifikator, identiteto odjemalca in števec ali naključno število [2]. Ta transakcija se nato izvede na vseh odobritvenih vrstnikih, ki glede na lastno shranjeno trenutno stanje ustvarijo „readset“, ki je sestavljen iz ključev in njihovih vrednosti, prebranih pred izvedbo, ter „writeset“, ki je sestavljen iz ključev in njihovih novih vrednosti [28]. Odjemalcem ti vrstniki nato vrnejo podpisano odobritev, ki vsebuje readset in writeset.

S tem se zaključi izvajalni del toka, čemur sledi urejevalni del, ki ga sproži odjemalec s oddajo vseh potrebnih odobritev, specificiranih v politiki odobravanja, urejevalniku. Ta poskrbi za ureditev transakcij v bloke, ki jih nato razpošlje po kanalu, na katerem se izvaja verižna koda. Urejevalniška storitev tu poskrbi, da pride do soglasja glede vrstnega reda transakcij na omrežju.

Kot zadnji del toka se izvede še potrjevanje blokov na vrstnikih. Ti najprej preverijo politiko odobravanja za vse transakcije. Če transakcija ni potrjena, vseeno ostane zapisana v bloku za namene hrambe zgodovine, njene spremembe pa ne spremenijo stanja glavne knjige. Nato vrstniki primerjajo trenutne vrednosti ključev z vrednostmi v readsetu. V primeru, da se vrednosti ne ujemajo, transakcije veljajo za neveljavne, saj je verjetno prišlo do sprememb vrednosti v času med izvedbo in potrjevanjem transakcije. Kot zadnji korak vrstniki nato prejeti blok pripnejo na konec verige in zamenjajo vrednosti v trenutnem stanju z vrednostmi iz writeseta [2].

3.2 Hyperledger Cello

Hyperledger Cello je skupek orodij pod okriljem projekta Hyperledger za avtomatizacijo postavitve, nadzora in upravljanja z omrežjem verig blokov. Storitev postavitve omrežja podpira različne infrastrukture: fizični (bare-metal) gostitelj, virtualna okolja ter več vsebniških platform [29]. Prav tako omogoča postavitev omrežja za povezovanje vsebnikov Docker, bodisi z Docker Swarm ali Kubernetees. Projekt je trenutno v inkubacijski fazi in podpira

predvsem ogrodje Hyperledger Fabric.

Cello uporabnikom (administratorjem) ponuja uporabniški vmesnik za upravljanje z lastnimi omrežji v obliki spletne strani. Ta poenostavi proces postavitve omrežij Hyperledger Fabric in zmanjša količino potrebnega dela za postavitev, nadzor ter spreminjanje omrežij. Tu lahko administrator enostavno postavi svoje omrežje ter nato spremlja njegovo stanje, spreminja število vozlišč in njegovo konfiguracijo [30]. Te prednosti so opazne predvsem, ko upravljamo z večjim številom omrežji, ki si lahko tudi delijo gostitelje.

3.3 Hyperledger Composer

Hyperledger Composer je skupek orodij za enostavnejše razvijanje poslovnih omrežij [31], kakor imenujemo aplikacije, napisane v tem orodju. Omogoča boljši pregled in nam ponuja avtomatizacijo upravljanja z omrežjem, kar pospeši razvoj aplikacij. Uporabnik za uporabo Composerja ne potrebuje veliko znanja o izvajanju verižne kode in se lahko osredotoča predvsem na implementacijo funkcionalnosti, ki jih želi implementirati. Za razvijalca oziroma administratorje Composer ponuja boljšo uporabniško izkušnjo, saj lahko večino dela opravijo preko grafičnega vmesnika namesto ukazne vrstice.

Hyperledger Composer definiranje aplikacije razbije na štiri dele, vsak pa je opisan v svoji datoteki. V datoteki s končnico *.cto* se nahaja opis modela. V njej v posebni sintaksi definiramo strukturo v obliki objektov, deljenih na sredstva, koncepte, udeležence in transakcije v našem poslovnem omrežju [32]. Tem določimo tudi lastne atribute in njihove tipe, ključ ter definiramo medsebojne odnose med objekti.

V datoteki s končnico *.js* nato definiramo funkcije za transakcije, ki smo jih definirali že v prejšnji datoteki, z uporabo jezika Javascript in podane knjižnice. Te transakcije definirajo premikanje sredstev po omrežju in njihov življenjski cikel med izmenjavo [3].

V datoteki *.acl* lahko v definirani sintaksi podrobno določimo pravila za dostope. V opisu udeležencem definiranim v datoteki *.cto*, določimo dovolje-

nja za izvajanje operacij nad privzetimi objekti ali pa objekti in transakcijami, ki smo jih že predhodno definirali.

Zadnja vrsta datoteke pa je *.qry*, kjer lahko napišemo poizvedbe za pregled trenutnega stanja poslovnega omrežja [32]. Te napišemo jeziku, ki je precej podoben SQL, kasneje pa jih kličemo s potrebnimi parametri in dobimo rezultat, ki opisuje trenutno stanje, zapisano v glavi knjigi [32]. Takšno omrežje Hyperledger Composer izvozi v datoteko *.bna* (Bussiness Network Archive), ki se jo nato lahko shrani ali prenaša in namesti na željeno omrežje Hyperledger Fabric. Če se Composer lahko poveže z omrežjem Hyperledger Fabric, pa se lahko takšno omrežje namesti kar preko lastnega grafičnega vmesnika, kar močno pospeši predvsem razvoj in testiranje poslovnih omrežij [3].

Poglavje 4

E-glasovanje

Izraz e-glasovanje ima več pomenov. Uporablja se za opis elektronskih glasovanj, kjer lahko govorimo o glasovanju preko fizičnih elektronskih naprav lociranih na voliščih ali pa kot oddaljeno glasovanje preko interneta, ki ni fizično nadzorovano [14]. Mi z izrazom opisujemo slednjega.

Digitalizacija glasovanj potencialno prinaša veliko prednosti pred klasičnim glasovanjem, pri katerem volivci oddajajo fizično označene glasovnice. Takšna transformacija procesa namreč lahko zmanjša stroške volitev, še posebej ko govorimo o volitvah na državni ravni. Prav tako bi elektronsko glasovanje poenostavilo pristop k volitvam, kar bi dvignilo udeležbo skupin volivcev, ki se trenutno volitev udeležujejo v manjšem številu, kot na primer mladi ali fizično ovirani [7]. Otežilo bi se tudi goljufanje pri procesu, kar bi še dodatno pripomoglo k transparentnosti, ki bi jo obljubljalo digitalno glasovanje, pri katerem bi bili vsi glasovi javni, ampak seveda ne povezljivi z volivci. Če oziroma ko bomo dosegli takšno stopnjo napredka pri digitalizaciji procesa, lahko to vidno in pozitivno vpliva tudi na samo delovanje demokratične države.

Kljub pozitivnim obljubam digitalnih volitev pa je ta proces eden zadnjih, ki se ga digitalizacija še ni dotaknila. Razlog je v tem, da takšne rešitve ne izpolnjujejo vseh načel, ki so potrebna za izvajanje volitev, in ponavadi sprejemajo manjše kompromise.

4.1 Načela

Za pravičen proces glasovanja se morajo volitve držati določenih načel. Eno izmed teh je volilna svoboda, ki zahteva, da pri procesu ni nasilja, zatiranja, ustrahovanja, pritiskov oziroma vplivov s strani organizatorja volitev ali kateregakoli posameznika. Tu je lahko problematičen tako vpliv, preden je glas oddan, kot na primer grožnje ali manipulacija z volivcem, kot tudi posledice, ki bi lahko sledile, če glasovanje ni bilo anonimno. Ta problem lahko rešimo z nesledljivostjo glasu s strani volivca. Tako onemogočimo izsiljevalcu ali osebi, ki je kupila glas, da preveri izbiro izsiljevanega volivca [10]. To je sicer prav tako lahko dvorezen meč, saj zdaj volivec ni zmožen preveriti, ali je bil njegov glas veljaven oziroma spremenjen. Ta pogoj je sicer zadoščen pri tradicionalnem glasovanju, pri elektronskem glasovanju pa je temu težje ugoditi oziramo moramo sprejeti odločitev, ali smo pripravljeni sprejeti kompromis. Svoboda govora je lahko kršena tudi v primeru, ko bi bilo v delu aplikacije za glasovanje prisotno pristransko sporočilo, ki bi napeljevalo k določeni izbiri. Prav tako mora glasovanje omogočati svobodo oddaje neveljavnega oziroma praznega glasu [7].

Pri glasovanju je pomembna tudi tajnost. Ko volivec odda glas, ne želimo, da lahko kdorkoli vidi njegovo izbiro, saj bi lahko to negativno vplivalo na svobodo izbire. Tu ni pomembna le dejanska tajnost samega glasovanja, temveč tudi psihološka tajnost, saj se v primeru, ko volivci ne verjamejo v poštenost organizacije, ki vodi volitve, to odraža pri njihovi izbiri [5]. Skrb za konsistentno integriteto je pomembna naloga organizacije, saj lahko že najmanjše nepravilnosti povzročijo dolgoročne posledice pri zaupanju volivcev in njihovih glasovih. Za primer – malo je dokazov, da glasovanje na predsedniških volitvah v ZDA ni tajno, a kljub temu okoli četrtina ljudi o tem dvomi [5]. Če želimo doseči čim večje zaupanje v proces, moramo poskrbeti za njegovo transparentnost in o postopku poučiti vse udeležence. Tajnost je še posebej težko zagotoviti pri elektronskem glasovanju, saj glasujemo na javnem omrežju in komunikacija poteka prek posrednikov. Ta problem se sicer pojavlja že pri glasovanju po pošti, kjer problem rešujemo na ravni

zakonodaje in regulacije.

Za zagotavljanje pravičnosti pri volitvah moramo poskrbeti tudi za enakost. Tu govorimo o enakosti volivcev, kot tudi o enakosti kandidatov oziroma volilnih izbir. Pri enakosti volivcev moramo poskrbeti, da je vsak glas enakovreden, razen ko je to drugače posebej definirano, na primer glasovanje delničarjev z različnimi vrstami delnic, katerih teže pri glasovanju se razlikujejo. Primer neenakosti, ki bi se lahko hitro zgodil ob izključno elektronskem glasovanju, je tudi neupoštevanje zmožnosti dostopa do elektronskih naprav oziroma pomanjkanje znanja o njih. Pri takšnem glasovanju bi namreč bil otežen dostop do glasovanja starejši populaciji, ki ni tako tehnološko pismena. Ta problem lahko delno rešimo s sistemom, ki je zelo intuitiven in uporabniku prijazen ter vsebuje navodila in pomoč na vsakem koraku postopka. Ko pa govorimo o enakosti med volilnimi izbirami, moramo poskrbeti, da naš sistem ne vsebuje pristranskosti oziroma ni naklonjen določeni izbiri. Kršitev pri elektronskem glasovanju bi v tem primeru lahko bila pristranskost strani, kot je na primer preferenčna postavitev izbire, oziroma da bi bila katera izmed izbir izrecno poudarjena [7].

Za uspešno demokratično glasovanje morajo naše volitve izpolnjevati še nekaj pogojev. Ker implementiramo elektronsko glasovanje, se moramo zavedati, da bo proces kompleksnejši in težje razumljiv za povprečnega volivca. Ta mora imeti veliko zaupanje v samo tehnologijo, kot tudi v ljudi, ki s sistemom operirajo oziroma prirejajo volitve. To pomeni, da moramo pri implementaciji sistema posvečati izjemno pozornost transparentnosti sistema na vseh delih procesa in da moramo že vnaprej razmisliti, kako sistem poenostaviti, saj mora biti čim bolj razumljiv za povprečnega volivca. Transparentnost lahko pripomore tudi k reševanju problema varnosti. Ker gre za elektronski sistem, vedno obstaja možnost, da nekje obstaja luknja v sistemu, ki omogoči manipulacijo s procesom. Seveda odpravljanju teh težav posvetimo zelo visoko pozornost in vpeljemo postopke testiranja, a sistem vseeno ni nikoli povsem varen. Zato je pomembno, da je proces transparenten, saj je v primeru, da pride do zlorabe, to lahko hitro opaženo in sledljivo,

kar omogoči sprožitev postopka za popravitev škode oziroma ponovitev volitev po odpravi napake v sistemu. Tehnologija prinaša tudi nekaj dodatnih orodij, s katerimi lahko povečamo zaupanje volivcev. Npr. beleženje vseh aktivnosti in uporabniku prijazen prikaz trenutnega stanja njegovega glasu in volitev. Sem lahko prištejemo tudi zmožnost preverjanja uporabnikove izbire in pregled statusa oddanega glasa v smislu, ali je ta že bil preštet in upoštevan. To je sicer lahko problematično s stališča svobode glasovanja, saj se zdaj lahko na volivca izvaja pritisk, da dokaže svoj glas [7].

4.2 V praksi

Elektronsko glasovanje je še vedno v povojih. Kljub poskusom mnogih držav je po naših podatkih Estonija trenutno edina država, ki na državnih volitvah omogoča elektronsko glasovanje. Ta je že leta 2005 volivcem omogočila oddajo elektronskega glasu na lokalnih volitvah, leta 2007 pa celo na parlamentarnih volitvah na državni ravni [1]. Priljubljenost takšnih volitev je s časom naraščala in leta 2017 je na lokalnih volitvah v Estoniji elektronsko glasovalo že več kot 31 % vseh volilnih udeležencev [36]. Za elektronsko glasovanje morajo udeleženci glasovati 6 do 4 dni pred volitvami, če pa se volivec poleg oddanega elektronskega glasu volitev udeleži tudi osebno, se njegov elektronski glas označi za neveljavnega.

V Estoniji za dokazovanje identitete uporabnikov uporabljajo elektronske osebne izkaznice, ki vsebujejo par ključev in so obvezne za vse državljane. Za avtentikacijo pri elektronskem glasovanju volivci potrebujejo za to namenjen elektronski čitalec, v katerega umestijo svojo osebno izkaznico in vnesejo prvi PIN. Nato izberejo kandidata, vnesejo drugi PIN in tako uspešno oddajo svoj glas [1]. Ena glavnih prednosti Estonije je ravno razširjen sistem digitalnih identitet. Njihova različica je implementirana v mnogih drugih državah, a je redko obvezna in morda izdana le v obliki digitalnih certifikatov, ki so za povprečnega volivca manj razumljivi od fizičnih.

Trenutno v široki uporabi še ni tehnologij, ki bi implementirale glasovanje

na verigi blokov. Kljub pridobitvi nekaterih pomembnih lastnosti, kot je na primer nespremenljivost verige, pa druge težave še vedno ostajajo. Čeprav celovite rešitve še ni, pa je v razvoju veliko projektov, ki za implementacijo uporabljajo različne pristope. Enega izmed takšnih projektov je implementiralo podjetje Nasdaq v Estoniji za borzo v Talinu, ki je omogočilo glasovanje delničarjem, pri čemer so za izvajanje pametnih pogodb uporabili tehnologijo Chain. Tu so za avtentikacijo uporabili že zgoraj opisane osebne izkaznice, kar je olajšalo proces, in projekt je bil razglašen za uspešnega [37].

Poglavje 5

Aplikacija za e-volitve

V sklopu te naloge smo postavili tudi celotno porazdeljeno omrežje Hyperledger Fabric za potrebe glasovanja znotraj podjetij. Aplikacija se uporablja v produkcijske namene, zaradi česar je stabilnost ključnega pomena. Za izpolnitev te zahteve smo porabili veliko časa tudi za enostavno in učinkovito postavitev porazdeljenega omrežja postavljenega v HA. Poleg omrežja sta sledili še implementaciji poslovnega omrežja, kot rečemo aplikaciji, napisani v Hyperledger Composerju, in uporabniškega vmesnika, ki je v našem primeru spletna aplikacija. Te tri sklope smo ločili in jih podrobneje opisali v naslednjih poglavjih.

5.1 Omrežje

Omrežje smo postavili kot visokorazpoložljivo, saj aplikacija teče v produkcijskem okolju, kar pomeni, da mora biti odporna na določene izpade. Omrežje smo postavili na štirih različnih gostiteljih, kar omrežju omogoča odpornost na izpad enega gostitelja. Zaradi potrebe po dopuščanju možnosti izpada smo za tip urejevalniških storitev izbrali način kafka, ki pa za CT potrebuje vsaj štiri vozlišča.

Ker se pri velikem številu komponent hitro pojavijo težave pri ročni konfiguraciji vseh vozlišč in ker smo omrežje želeli postaviti na več okoljih,

smo omrežje postavili s Hyperledger Cello. Iz tega orodja smo uporabili izključno del Ansible, pri tem pa smo morali skripto, dostopno preko `https://github.com/oberzan/cello`, prilagoditi lastnim potrebam. To smo storili, ker smo želeli avtomatizirati tudi postavitev Hyperledger Composerja in vse certifikate postopoma ustvariti z lastnim korenskim CA, in ne z orodjem *cryptogen*. Skripta nam zdaj omogoča, da lahko popravimo le nekaj spremenljivk v konfiguracijski datoteki, in omrežje se avtomatsko postavi na v njej določenih gostiteljih.

Skripta je sestavljena iz dveh delov. V prvem delu se za naše omrežje pripravi okolje in poskrbi, da se na vseh gostiteljih namestijo vsa potrebna orodja, programski jezik Go, ki je potreben za izgradnjo binarnih datotek Fabric, ter da se uredijo sistemske nastavitve. Ker za naše omrežje uporabljamo gostitelje z operacijskim sistemom CentOS, je bilo potrebnih več sprememb, saj ta za upravljanje paketov ne uporablja orodij APT, ampak Yum. Nato se namesti porazdeljen KVS *etcd*, *flannel*, ki poskrbi za povezljivost med vsemi vozlišči v našem omrežju in nazadnje še vsebnik SkyDNS, ki deluje kot DNS in nam omogoča naslavljanje vsebnikov Docker z lastnimi imeni znotraj drugih vsebnikov v našem omrežju. Shrambo etcd uporabljata flannel in SkyDNS, pri čemer ga prvi uporablja za hrambo konfiguracije omrežja in dodeljenih podomrežij, slednji pa vanj hrani vse preslikave med naslovi in imeni vsebnikov, ki jih uporablja DNS. Po koncu izvajanja te skripte je naše okolje pripravljeno na namestitev komponent Hyperledger Fabric.

Druga skripta poskrbi za namestitev celotnega omrežja Hyperledger Fabric. Najprej se na vseh gostiteljih ustvari direktorij za naš projekt, v katerega se že takoj zatem prenesejo in zgradijo binarne datoteke orodij Fabric, kot sta *configtxgen* in *configtxlator*, ter se prenesejo *Docker* slike za Fabric CA. Nato se na izbranem gostitelju postavi korenski CA, ki si ob inicializaciji ustvari samopodpisani certifikat, zatem pa se glede na konfiguracijo postavijo še vmesne CA, na primer za vsako organizacijo posebej. Te CA ob inicializaciji za svoj certifikat zaprosijo korenski CA, ki preveri poverilnice in izda certifikat, ki lahko sam podpisuje in izdaja certifikate. Korenski CA se

lahko nato ugasne, saj ga razen v izjemnih primerih, ko želimo dodati nov vmesni CA, ne potrebujemo več.

Ko imamo postavljene certifikatne avtoritete, lahko začnemo z registracijo vseh entitet v našem začetnem omrežju. Najprej se odjemalec s privzetimi poverilnicami prijavi v CA vseh urejevalniških organizacij ter zanje registrira lastne urejevalnike in njihove administratorje. Nato sledi še prijava v vse CA vseh drugih organizacij, kjer se registrira vse vrstnike, njihove administratorje ter testne uporabnike. Ko smo ustvarili vse zgoraj opisane certifikate, jih še uredimo v skupen direktorij za lažje dostopanje do potrebnih poverilnic znotraj vseh vsebnikov Docker.

Po zaključenem delu s certifikati iz priloženih predlog sestavimo konfiguracijske datoteke za vrstniška vozlišča in konfiguracijsko datoteko (*configtx.yaml*) za privzeti kanal, ki ga bomo ustvarili. Sedaj lahko z orodjem Hyperledger Fabric *configtxgen* ustvarimo genezni blok za sistemski kanal urejevalnikov in transakcijo za stvaritev privzetega kanala, katerega ime je določeno v konfiguracijski datoteki. Nato za vsako organizacijo z orodjem *configtxgen* ustvarimo še sidrne vrstnike in genezni blok ter vse transakcijske datoteke prenesemo na vse gostitelje.

V naslednjem koraku s strani Docker Hub naložimo vse potrebne slike ter za vrstniška, urejevalniška, Kafka in Zookeeper vozlišča iz predlog ustvarimo docker-compose datoteke. Za aplikacijo smo tu določili postavitev štirih vozlišč Kafka, za zagotavljanje odpornosti na izpad enega vozlišča in postavitev treh instanc Zookeeper, od katerih je odvisna Kafka. Število 3 nam tu predstavlja najmanjše liho število, ki je pogoj za CT, saj je Zookeeper porazdeljen KVS, ki za delovanje potrebuje liho število vozlišč, da se izognemo problemu razcepljenih možganov [33]. Za Pogoje CT postavimo še dve instanci urejevalnikov in vrstnikov. Vsak vrstnik tu uporablja svojo zunanjo bazo CouchDB, kar izboljša pregled nad trenutnim stanjem omrežja in poenostavi poizvedovanje po trenutnem stanju glavne knjige. Zadnja izmed komponent, ki jo postavimo, pa je vsebnik z orodji Hyperledger Fabric, saj imamo tako orodja na voljo brez izgradnje iz binarnih datotek in pri

upravljanju s poverilnicami ne spreminjamo datotek na gostitelju.

Nato z docker-compose dvignemo vse zgoraj definirane vsebnike in omrežje se postavi. Zdaj moramo še definirati naš kanal, po katerem bo potekala komunikacija naše aplikacije. To storimo tako, da z enega vrstnika s priloženo transakcijo, ki smo jo predhodno ustvarili, na urejevalnik pošljemo zahtevek za ustvaritev kanala in podatek o imenu kanala. Nato moramo vse vrstnike še pridružiti v kanal in ga posodobiti s transakcijo, ki opisuje sidrne vrstnike.

5.2 Pametna pogodba

Celotno poslovno omrežje smo definirali z uporabo Hyperledger Composerja. To nam je omogočilo ločitev dela definicije vseh entitet, njihovih dovoljenj in funkcij na posamezne dele. Prav tako Hyperledger Composer ponuja orodje composer-rest-server, ki avtomatizira postavitev REST API-ja. Orodju podamo konfiguracijske podatke, vključno s poverilnicami željenega uporabnika, ta pa nam postavi strežnik, ki izpostavlja naše poslovno omrežje preko API-jev. Te generira z ogrodjem LoopBack in jih opiše z dokumentacijo Swagger. Tako se enostavno izognemo zahtevi, da s poslovnim omrežjem komuniciramo preko SDK-ja.

Celotno poslovno logiko bi lahko definirali tudi v izvirni verižni kodi, a bi bilo precej odvečnega dela, ki bi hkrati predstavljalo nov prostor za napake. Shramba verižne kode namreč uporablja KVS, kamor lahko za izbrani ključ shranimo poljubni objekt. To pomeni, da moramo sami poskrbeti za pravilno manipuliranje z našimi objekti. Pri definiciji funkcij ne bi prišlo do velikih sprememb, bi pa bilo treba posebej definirati logiko omejevanja dostopov. Hyperledger Composer to funkcionalnost podpira z izdajanjem poverilnic udeležencem in nato omejevanjem dostopa z logiko, opisano v dostopnem seznamu. V tem primeru bi morali tudi vso komunikacijo speljati preko Hyperledger Fabric SDK-ja, kar pa spet pomeni dodatno odvečno delo in možnost za nove napake. Poudarek na preprostosti je pomemben, saj želimo doseči čim večjo transparentnost.

V tem poslovnem omrežju smo definirali dva modela. Prvi predstavlja glasovanje, ki ga kreira administrator in vsebuje svoj naslov kot identifikator. Kot atribute vsebuje tudi opis glasovanja, konec glasovanja, začetek glasovanja, ki se nastavi, ko administrator objavi žetone, in seznam modelov glasov, ki pripadajo temu glasovanju. Drugi model predstavlja glas glasovanja in ima za atribute razpršeni žeton, ki je tudi njegov identifikator, referenco na glasovanje, kateremu pripada, in opcijska atributa, ki vsebujeta žeton in izbiro. Ta sta neobvezna, saj se objekt glasu ustvari ob kreiranju glasovanja, ko ta atributa še nimata vrednosti, ki pa se nastavijo, ko volivci oddajo svoj glas. Poleg teh dveh modelov poslovno omrežje definira tudi dve funkciji. Prva se imenuje *publishTokens*. To funkcijo kliče administrator, ki omogoči glasovanje za vse žetone, ki so bili poslani volivcem. Kot vhodni podatek prejme nabor vseh razpršenih žetonov in jih shrani k podanemu glasovanju. Po izvedbi funkcije se začne glasovanje, saj je za žetone zdaj objavljen njihov par. Kdor želi glasovati, mora poleg svoje izbire poslati tudi žeton, za katerega mora biti za te volitve že predhodno objavljena njegova razpršena vrednost. Ker je vhodno vrednost zgoščevalne funkcije (v našem primeru žeton) praktično nemogoče izpeljati iz njene izhodne vrednosti, je edini, ki lahko glasuje, tisti, ki pozna vhodni žeton, to pa je volivec, kateremu je žeton poslan. Za glasovanje pa je definirana funkcija z imenom *publishVote*, ki za vhodne podatke prejme identifikator volitev, izbrano glasovalno izbiro in uporabniku dodeljen žeton. Funkcija ob prejetem klicu razprši podani žeton, in če je za podane volitve ob začetku že bil objavljen identični razpršeni žeton, se v model glasu pripiše podana izbira.

Tretji del definicije naše pametne pogodbe je definiranje seznama dostopov. Ker smo se odločili za glasovanje s podajanjem žetonov in ker uporabniki ne bodo imeli lastnih poverilnic za dokazovanje identitete, ki bi lahko ogrozile njihovo zasebnost, je ta seznam zelo enostaven. Tu podamo le polni dostop administratorski identiteti, katerega poverilnice imamo na strežniku aplikacije, s katero komuniciramo s poslovnim omrežjem.

5.3 Uporabniški vmesnik

Za uporabniški vmesnik smo implementirali spletno aplikacijo, katere koda je dostopna preko `https://github.com/oberzan/cello`. Ta v zaledju uporablja zgoraj opisano poslovno omrežje, in je vmesnik za komuniciranje z verižno kodo, ki teče na Hyperledger Fabric, s katero aplikacija komunicira preko ponujenega SDK-ja. Kot že pri implementaciji poslovnega omrežja, je bil eden naših glavnih ciljev, da implementiramo kar se da enostaven in intuitiven uporabniški vmesnik. Tu se osredotočamo predvsem na uporabniški del, saj s tem povečamo transparentnost procesa.

Strežnik smo implementirali v tehnologiji NodeJS s pomočjo ogrodja Express. Ta streže spletne strani, ki smo jih podrobneje opisali v podpoglavjih, in ponuja API-je, ki jih v zaledju uporabljajo servirane spletne strani. Ti nam omogočajo, da za komunikacijo z verižno kodo ni treba ponovno nalagati spletne strani, kar bi bilo precej moteče predvsem pri brisanju podatkov v obrazcih. Za pošiljanje elektronskih sporočil smo implementirali tudi povezavo na strežnik SMTP s knjižnico Nodemailer. Alternativno smo za pošiljanje elektronskih sporočil implementirali še pošiljanje sporočil SMS, kjer se strežnik v ozadju poveže na prehod SMS, s katerim smo komunikacijo vzpostavili s klici SOAP.

Spletne strani za enostavnejše oblikovanje vmesnika uporabljajo ogrodje Bootstrap, njihova vsebina pa je razdeljena na dva dela – na administrativni in uporabniški del.

5.3.1 Administrativni del

Administrativni del aplikacije je namenjen skrbniku omrežja za glasovanje, da lahko ustvari in upravlja z obstoječimi volitvami. Za dostop do administrativnega zaslona administrator obišče stran */admin*, pri čemer je ob prvem dostopu preusmerjen na */authenticate*, kjer se lahko prijavi. To je zelo preprosta stran z enim vnosnim poljem za vpis enotnega administrativnega gesla, ki je nastavljeno v konfiguracijski datoteki, ki jo uredimo pred

zagonom strežnika. Ko administrator vpiše in odda administrativno geslo, je preusmerjen na stran */admin*. Administratorju je tu sedaj omogočen dostop, saj je ob uspešni avtentikaciji prejel piškotek JWT s časovno omejeno veljavnostjo. Aplikacija ob prejemu tega žetona prebere čas poteka in ga shrani v shrambo seje za lažje kasnejše naslavljanje.

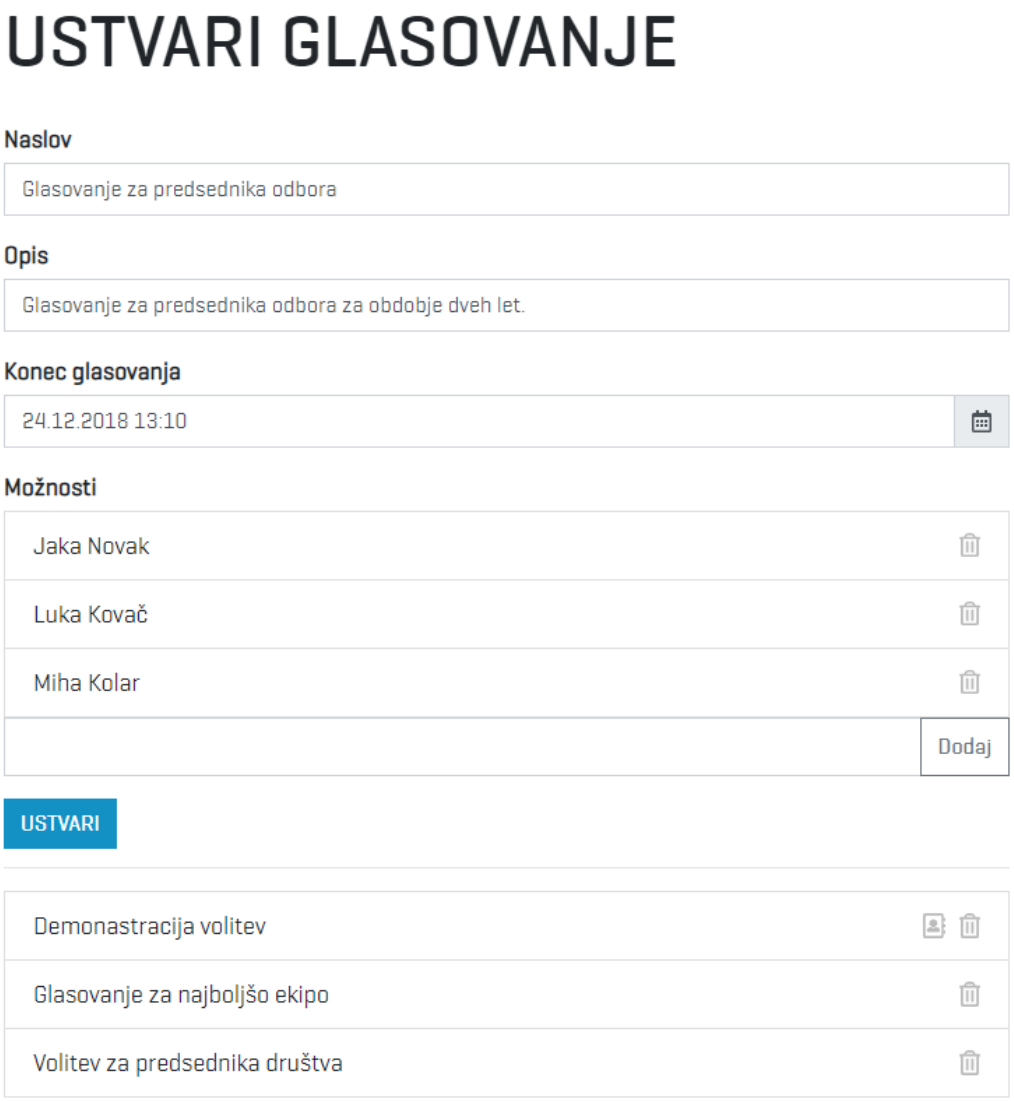
Na administrativni strani, prikazani na sliki 5.1, je na vrhu obrazec za vnos novega glasovanja.

Stran smo zasnovali tako, da ob oddaji zahtevka za kreiranje volitev že lokalno preverimo pravilnosti vnosa podatkov, kot je pogoj, da je datum konca volitev nastavljen v prihodnosti, in da sta na voljo vsaj dve izbiri za glasovanje. Na administrativnem zaslonu je pod obrazcem tudi seznam vseh že obstoječih volitev. Ob vseh volitvah sta na desni strani gumb za izbris volitev in pogojno, če se volitve še niso začele, gumb za začetek volitev.

Gumb za izbris volitev je namenjen predvsem za čiščenje volitev, ki so bile preklicane oziroma niso več pomembne za administratorja, ki potrebuje boljši pregled nad trenutnimi volitvami. Ob kliku se v pametni pogodbi označi, da so volitve preklicane, podatki pa ostanejo v distribuirani glavni knjigi, kar je eden izmed razlogov za njeno uporabo.

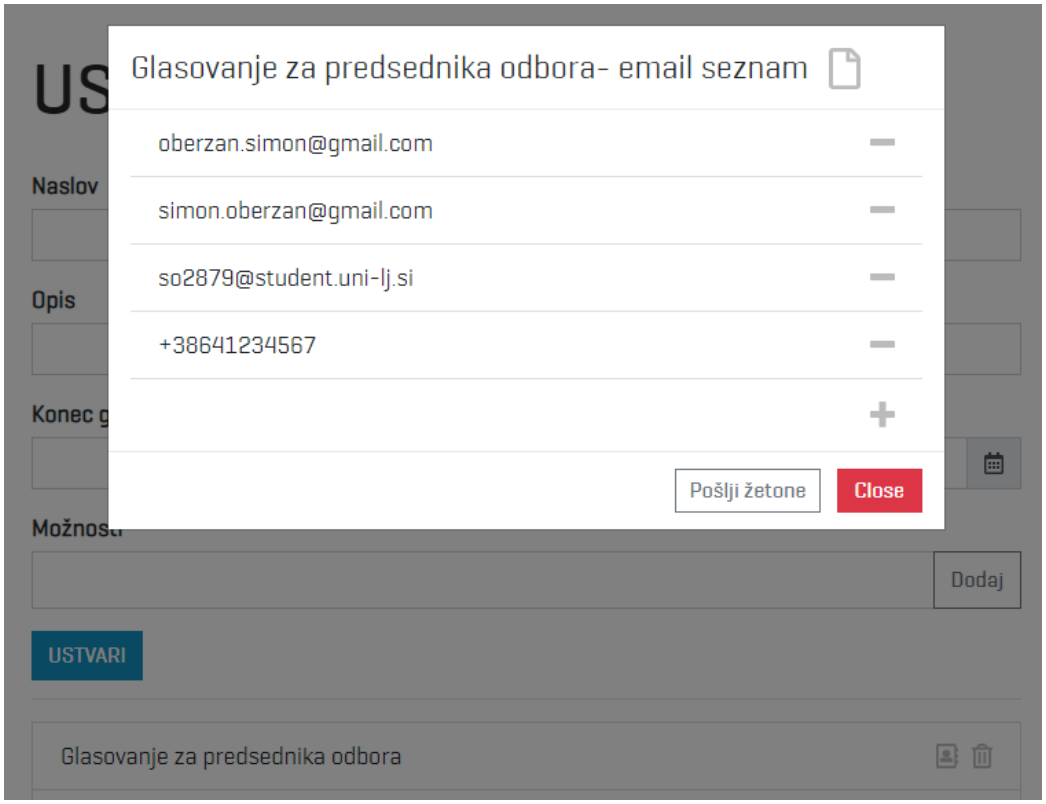
Ko administrator klikne na gumb za izbiro udeležencev volitev, se mu najprej odpre modalno okno, prikazano na sliki 5.2, kjer lahko ročno doda elektronske naslove oziroma mobilne številke vseh volivcev. Ko administrator uredi seznam vseh volivcev, lahko na dnu modalnega okna klikne na gumb za pošiljanje žetonov. Ob kliku na ta gumb se na strežniku ustvari naključni unikatni žeton za vsakega podanega volivca. Te žetone se nato razprši in objavi na verigo blokov v atribut našega glasovanja, uporabnikom pa se pošlje sporočilo in izvirni žeton na njihov e-poštni naslov oziroma se jim pošlje sporočilo SMS.

Za zagotovitev večje varnosti administratorju ob prijavi izdamo žeton s krajšo življenjsko dobo. Ker bi bila preusmeritev administratorja ob neuspešnem oddanem zahtevku izredno moteča, predvsem zaradi potrebne ponovne izpolnitve obrazca, smo implementirali modalno okno za podaljšanje



Slika 5.1: Prikaz kreiranja novih volitev na zaslonu */admin*. Obrazec vsebuje polja za vpis naslova, opisa glasovanja, datuma in časa, ki opisujeta, kdaj se bo glasovanje zaključilo, vnos množice vseh mogočih izbir za glasovanje in gumb, s katerim ustvarimo volitve za vpisane podatke.

vpisa. Ko se stran naloži, ta preveri čas poteka žetona in nastavi časovnik sprožitve prijavnega modalnega okna. Tu lahko administrator preprosto ponovi vpis enotnega gesla in ob uspešni prijavi nadaljuje z delom.



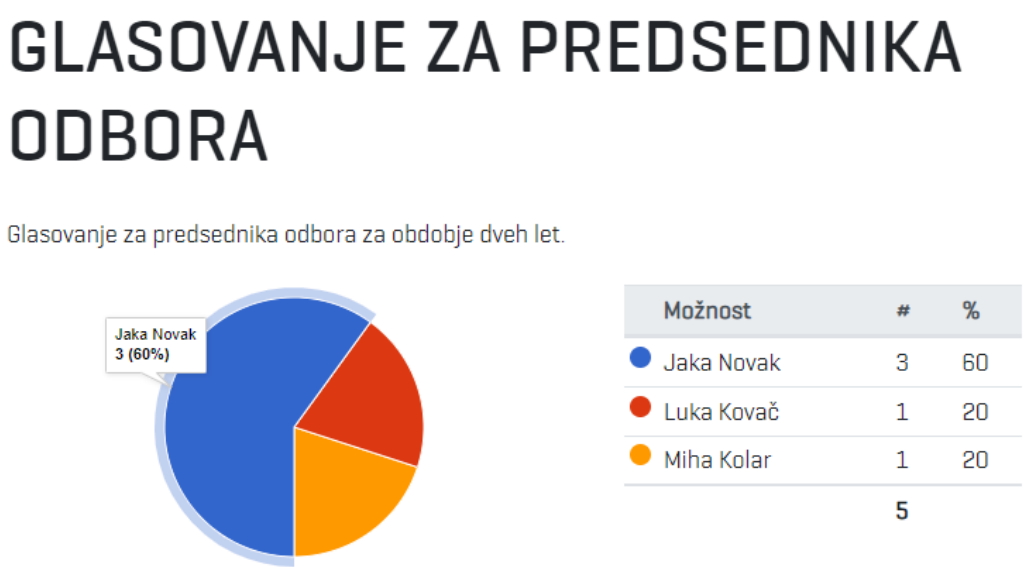
Slika 5.2: Prikaz vnosa volivcev in pošiljanja žetonov. Za lažje upravljanje večjega števila udeležencev je na vrhu okna tudi gumb, s katerim lahko seznam udeležencev uvozimo iz datoteke, ki vsebujejo naslove, ločene s presledki ali vejicami.

5.3.2 Uporabniški del

Za uporabnika se volitve začnejo ob prejemu sporočila o začetku volitev. Udeleženec je v sporočilu obveščen o naslovu volitev in ima poleg žetona priloženo še povezavo do strani za glasovanje. Povezava, ki je posredovana uporabniku, je sestavljena iz povezave do volitev (stran se nahaja na poti z imenom volitev, npr. */volitve x*) in iz atributa z vrednostjo žetona. Uporabnik lahko obišče stran glasovanja in ročno vnese žeton v za to namenjeno polje oziroma sledi prejeti povezavi, ki to zanj naredi samodejno, saj je žeton podan v atributu URL-ja. Poleg polja za žeton so na tej strani uporabniku

prikazani tudi naslov volitev, opis in seznam vseh izbir. Njihovo zaporedje je naključno, saj želimo slediti načelu enakosti med volilnimi izbirami. Vrstni red se določi v brskalniku, kar uporabnikom omogoča, da lahko po želji vidijo algoritem in imajo zagotovilo, da je vrstni red izbire v procesu resnično naključen. Ko uporabnik izbere željeno izbiro, jo potrdi s klikom na gumb „Oddaj glas“. Po oddaji se stran osveži, in ker je v atributu URL-ja podan uporabnikov žeton, strežnik preveri stanje njegovega glasu in prepozna, da je bil zanj že oddan glas. Tu se uporabniku izpiše obvestilo o uspešni oddaji glasu in trenutnem izbranem glasu. Ko volivec po oddanem glasu zapusti stran, se lahko kadarkoli vrne na stran volitev in po vpisu žetona preveri svojo trenutno izbiro.

Po zaključku volitev se stran uporabnika iz obrazca za vnos glasu spremeni v povzetek o volitvah, kjer lahko kdorkoli pregleda rezultate. Kot je razvidno na sliki 5.3, smo jih prikazali v obliki tortnega diagrama, ki smo ga implementirali s pomočjo knjižnice Google Charts.



Slika 5.3: Prikaz rezultatov volitev.

Ta omogoča tudi samodejno generiranje in prikaz legende, a ta ni bila v skladu s slogom aplikacije, zato smo jo implementirali sami. Iz obeh prikazov

lahko vidimo število oddanih glasov za vsako izbiro, vključno z neoddanimi glasovi, in njihov skupni delež.

Poglavje 6

Sklepne ugotovitve

Kot smo spoznali, so volitve izredno občutljiv proces. Izpolnjevati morajo mnogo pogojev, če želimo, da bo glasovanje resnično demokratično in pošteno. Klasični pristop h glasovanju ima sicer več pomanjkljivosti, a moramo z alternativnimi rešitvami biti zelo pazljivi, saj si težko privoščimo kakršenkoli kompromis – takšno glasovanje bi moralo imeti le prednosti. A žal ni tako – naša rešitev ima svoj nabor pomanjkljivosti. Ena izmed njih je, da ima administrator pri trenutni implementaciji moč ugotoviti, kakšen glas so oddali udeleženci. Problem izhaja iz tega, da so žetoni ustvarjeni lokalno na strežniku aplikacije in nato odposlani udeležencem, pri čemer bi lahko administrator preko povezovanja žetona z naslovom posledično deanonimiziral udeleženca in pregledal njegovo izbiro. Kot možno rešitev za ta problem lahko za pošiljanje žetonov na seznam vseh naslovov zadolžimo neodvisno organizacijo. V vsakem primeru pa uporabniki potrebujejo zaupanje v sistem, saj je ta kompleksnejši od klasičnega. Pri rešitvi na tehnologiji blokov je potrebno zaupanje v administratorje, ki razpošljejo žetone, in ne v organizatorje volitev oziroma osebju, ki prešteva glasove. Tu moramo torej sprejeti kompromis. Naš sistem uporabnikom zagotavlja pošten proces, kjer lahko enostavno glasujejo in preverjajo svojo izbiro z možnostjo, da je njihova izbira lahko odkrita. Pri klasičnem sistemu pa imajo sicer zagotovilo o anonimnosti, a je vse štetje in odločitev o rezultatu prepuščena organizator-

jem.

Naš sistem bi lahko izboljšali s kriptiranjem žetonov, ki bi bili objavljeni na verigo blokov. Vsak volivec bi nato z zasebnim ključem lahko prebrali svoj žeton, s čimer bi se znebili možnosti povezovanja žetonov z volivci preko komunikacijskih kanalov. A te izbire nismo sprejeli, saj v našem primeru nimajo vsi volivci lastnih certifikatov oziroma parov ključev ali pa je upravljanje z njimi zanje prezahtevno. Če bi vsi volivci imeli svojo digitalno identiteto, kot v primeru Estonije, bi lahko proces izboljšali, a je v praksi to pogosto težko izvedljivo.

Literatura

[1] R. Michael Alvarez, Thad E. Hall in Alexander H. Trechsel. Internet Voting in Comparative Perspective: The Case of Estonia. *PS: Political Science & Politics*, 42(3): 497–505, 2009. DOI: 10.1017/S1049096509090787.

[2] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich in sodelavci. Hyperledger fabric: a distributed operating system for permissioned blockchains. V *Proceedings of the Thirteenth EuroSys Conference*, stran 30. ACM, 2018.

[3] Vikram Dhillon, David Metcalf in Max Hooper. *Blockchain Enabled Applications: Understand the Blockchain Ecosystem and How to Make it Work for You*. Springer, 2017.

[4] Nitin Gaur, Luc Desrosiers, Venkatraman Ramakrishna, Petr Novotny, Salman Baset in Anthony O'Dowd. *Hands-On Blockchain with Hyperledger: Building decentralized applications with Hyperledger Fabric and Composer*. Packt Publishing Ltd, 2018.

[5] Alan S Gerber, Gregory A Huber, David Doherty in Conor M Dowling. Is there a secret ballot? Ballot secrecy perceptions and their implications for voting behaviour. *British Journal of Political Science*, 43(1): 77–102, 2013.

[6] Arthur Gervais, Ghassan O Karame, Vedran Capkun in Srdjan Capkun. Is bitcoin a decentralized currency? *IEEE security & privacy*, 12(3): 54–60, 2014.

[7] Dimitris A Gritzalis. Principles and requirements for a secure e-voting system. *Computers & Security*, 21(6): 539 - 556, 2002. ISSN: 0167-4048. DOI: https://doi.org/10.1016/S0167-4048(02)01014-3. URL: http://www.sciencedirect.com/science/article/pii/S0167404802010143.

[8] Markus Knell in Helmut Stix. Trust in Banks - Evidence from normal times and from times of crises. eng. V številka A1-V1 v zbirki Beiträge zur Jahrestagung des Vereins für Socialpolitik 2010: Ökonomie der Familie - Session: Trusting Banks in a Financial Crisis. Verein für Socialpolitik, 2010. URL: http://hdl.handle.net/10419/37184.

[9] Philip Koshy, Diana Koshy in Patrick McDaniel. An analysis of anonymity in bitcoin using p2p network traffic. V *International Conference on Financial Cryptography and Data Security*, strani 469–485. Springer, 2014.

[10] Thomas W Lauer. The risk of e-voting. *Electronic Journal of E-government*, 2(3): 177–186, 2004.

[11] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. [Dostopano 22. 12. 2018].

[12] Melanie Swan. *Blockchain: Blueprint for a new economy*. "O'Reilly Media, Inc.", 2015.

[13] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151: 1–32, 2014.

[14] Dimitrios Zissis in Dimitrios Lekkas. Securing e-Government and e-Voting with an open cloud computing architecture. *Government Information Quarterly*, 28(2): 239–251, 2011.

Dokumentacija

[15] Dosegljivo: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs. [Dostopano 12. 1. 2018].

[16] Dosegljivo: https://https://www.hyperledger.org/members. [Dostopano 12. 1. 2019].

[17] Dosegljivo: https://en.wikipedia.org/wiki/Hyperledger#cite_note-2. [Dostopano 22. 12. 2018].

[18] Dosegljivo: https://hyperledger-fabric.readthedocs.io. [Dostopano 22. 12. 2018].

[19] Dosegljivo: https://hyperledger-fabric.readthedocs.io/en/release-1.4/peers/peers.html. [Dostopano 15. 1. 2019].

[20] Dosegljivo: https://hyperledger-fabric.readthedocs.io/en/release-1.4/fabric-sdks.html. [Dostopano 15. 1. 2019].

[21] Dosegljivo: https://hyperledger-fabric.readthedocs.io/en/release-1.4/channels.html. [Dostopano 12. 1. 2019].

[22] Dosegljivo: https://hyperledger-fabric.readthedocs.io/en/release-1.4/gossip.html. [Dostopano 15. 1. 2019].

[23] Dosegljivo: https://jira.hyperledger.org/browse/FAB-33. [Dostopano 12. 1. 2019].

[24] Dosegljivo: https://hyperledger-fabric.readthedocs.io/en/release-1.4/identity/identity.html. [Dostopano 14. 1. 2019].

[25] Dosegljivo: https://hyperledger-fabric.readthedocs.io/en/release-1.4/identity/identity.html. [Dostopano 14. 1. 2019].

[26] Dosegljivo: https://hyperledger-fabric.readthedocs.io/en/release-1.4/ledger/ledger.html. [Dostopano 15. 1. 2019].

[27] Dosegljivo: https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html. [Dostopano 15. 1. 2019].

[28] Dosegljivo: https://hyperledger-fabric.readthedocs.io/en/release-1.4/readwrite.html. [Dostopano 10. 1. 2019].

[29] Dosegljivo: https://www.hyperledger.org/projects/cello. [Dostopano 22. 12. 2018].

[30] Dosegljivo: https://hyperledger-cello.readthedocs.io/en/latest/. [Dostopano 15. 1. 2019].

[31] Dosegljivo: https://www.hyperledger.org/projects/composer. [Dostopano 3. 1. 2019].

[32] Dosegljivo: https://hyperledger.github.io/composer/latest. [Dostopano 28. 12. 2018].

[33] Dosegljivo: https://hyperledger-fabric.readthedocs.io/en/release-1.4/kafka.html. [Dostopano 26. 1. 2019].

[34] An Introduction to Hyperledger. Dosegljivo: https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf. [Dostopano: 13. 1. 2019].

[35] Vitalik Buterin. Slasher: A punitive proof-of-stake algorithm. Dosegljivo: https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/, 2014.

[36] Rezultati Estonskih lokalnih volitev 2017. Dosegljivo: https://kov2017.valimised.ee/valimistulemus-vald.html. [Dostopano 28. 12. 2018].

[37] Is Blockchain the Answer to e-Voting? Nasdaq Believes so. Dosegljivo: https://business.nasdaq.com/marketinsite/2017/Is-Blockchain-the-Answer-to-E-voting-Nasdaq-Believes-So.html. [Dostopano 16. 1. 2019].