Aplikacija za glasovanje koristeći blockchain tehnologiju

Stipanić, Tomislav

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni diplomski studij računarstva

APLIKACIJA ZA GLASOVANJE KORISTEĆI BLOCKCHAIN TEHNOLOGIJU

Diplomski rad

Tomislav Stipanić

Osijek, 2021.

SADRŽAJ

1. UVOD 1

1.1. Zadatak diplomskog rada 1

2. IZAZOVI ELEKTRONIČKOG GLASOVANJA I POSTOJEĆA RJEŠENJA 2

2.1. Elektroničko glasovanje na biračkim mjestima 2

2.2. Elektroničko glasovanje na daljinu 2

2.3. Blockchain glasovanje 3

2.4. Voatz 6

2.5. Idejno rješenje aplikacije za glasovanje 7

3. PRIMIJENJENE TEHNOLOGIJE I ALATI 9

3.1. Arhitektura 9

3.2. Ethereum 10

3.2.1. Solidity 11

3.2.2. Truffle & Ganache 11

3.1. Firebase 12

3.1.1. Autentikacija 12

3.1.2. Firestore baza podataka 12

3.2. Flutter 13

3.2.1. Domain Driven Design 13

3.2.2. Web3 14

4. PROGRAMSKO RJEŠENJE APLIKACIJE ZA GLASOVANJE 15

4.1. Pametni ugovor 15

4.2. Sloj podataka 18

4.2.1. Firebase Autentikacija 19

4.2.2. Firebase Firestore 20

4.2.3. Web3 21

4.3. Sloj domene 23

4.4. Sloj aplikacije 26

4.4.1. Stanje 27

4.4.2. Događaj 28

4.4.3. Upravljanje slučajevima 29

4.5. Sloj korisničkog sučelja 30

5. KORIŠTENJE APLIKACIJE ZA GLASOVANJE 33

5.1. Autentikacija korisnika 33

5.2. Kreiranje nove glasačke kutije 34

5.3. Glasovanje 36

5.1. Pregled prijašnjih glasačkih listića 37

6. ZAKLJUČAK 39

LITERATURA 40

SAŽETAK 42

ABSTRACT 43

ŽIVOTOPIS 44

PRILOZI 45

1. UVOD

Od samih početaka njegove popularizacije, blockchain tehnologija se pokušava primijeniti na različite probleme, ali i na već postojeća rješenja. Jedan od tih problema je i elektroničko glasovanje. Ideja iza sustava elektroničkog glasovanja na blockchainu je osigurati transparentan, siguran, povjerljiv i pristupačan izborni postupak. Blockchain tehnologija osigurava da svatko tko sudjeluje u izborima može provjeriti jesu li glasali za željenog kandidata šifriranjem, odnosno dešifriranjem svoga glasa. Aplikacija za ovakav način glasovanja bi omogućila slobodan, pristupačan i transparentan način glasovanja. No uz sve prednosti, takav sustav ima i ozbiljne nedostatke.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada je napraviti aplikacija za glasovanje koristeći blockchain tehnologiju. Potrebno je proučiti i trenutnu zakonsku regulativu glasovanja u državnom i privatnom sektoru. Za primjer uzeti glasovanje u nekoj udruzi ili klubu.

Kroz proces izrade aplikacije bit će objašnjen fenomen elektroničkog glasovanja i njegovi nedostaci. Drugo će poglavlje čitatelja upoznati o elektroničkom glasovanju pri čemu će odgovoriti na pitanje poboljšava li se funkcionalnost i sigurnost procesa elektroničkog internet glasovanja dodatkom blockchain tehnologije. Treće poglavlje govori o tehnologijama i alatima korištenim za razvoj aplikacije. Četvrto poglavlje opisuje arhitekturu i implementaciju programskog koda pri razvoju mobilne aplikacije za glasovanje pomoću blockchain tehnologije. U petom poglavlju su dane upute za korištenje gdje će čitatelj dobiti i uvid u sam izgled mobilne aplikacije.

2. IZAZOVI ELEKTRONIČKOG GLASOVANJA I POSTOJEĆA RJEŠENJA

Elektroničko glasovanje se pojavljuje u dva oblika:

- elektronički uređaji, locirani na biračkim mjestima koji su fizički nadzirani od vladinih udruga ili neovisnih izbornih tijela
- daljinsko glasovanje, gdje birač svoj glas predaje izbornom tijelu putem interneta s bilo koje lokacije

2.1. Elektroničko glasovanje na biračkim mjestima

Uređaj za elektroničko glasovanje, koji je vrlo popularni u Americi pod kraticom DRE (engl. *Direct-Recording Electronic*), je uređaj dizajniran za omogućavanje izravnog glasovanja na biračkim mjestima. Glasač glasuje preko ekrana osjetljivog na dodir te se njegov glas pohranjuje izravno na internu memoriju uređaja. Nakon kritika stručnjaka vezanih za nemogućnost revizije glasova, u kasnijim iteracijama se dodaje mogućnost fizičkog ispisa glasova. Pri takvom načinu glasovanja se izbjegavaju ponekada dvosmislene ili nejasne oznake na glasačkim listićima, te se kao prednosti ističe i mogućnost samostalnog glasovanja za osobe s invaliditetom.

Gotovo svako provedeno istraživanje o sigurnosti elektroničkih uređaja za glasovanje na biračkim mjestima je pronašlo ozbiljne probleme [1]. Kao veliki problem naglašava se nedovoljno ili nepotpuno korištenje kriptografije. Javljaju se problemi poput, nezaštićenih pohranjenih podataka, loše implementacija slučajnih brojeva, nezaštićenog mrežnog prometa i lošeg upravljanja kriptografskim ključevima. Primjerice na brazilskim uređajima za glasanje je u istraživanju otkriveno da svi glasački uređaji unutar iste regije dijele isti kriptografski ključ. Što bi omogućilo napadaču da se domogne izvornog koda jednog takvog stroja i ugrozi glasove milijunima ljudi diljem zemlje. Otkriveni su i problemi nevezani za kriptografiju, poput nedostatak kontrole pristupa i loše provjere ulaznih podataka. No, iako elektronički uređaji za glasovanje imaju svoju svrhu, primjenu i nedostatke, nisu izričito bitni u kontekstu blockchain glasovanja.

2.2. Elektroničko glasovanje na daljinu

Daljinsko elektroničko glasovanje definiramo kao glasovanje kojemu pristupamo preko interneta pomoću računala ili mobilnog uređaja. Drugim riječima, glasač svoj glas izbornom tijelu dostavlja elektroničkim putem. Takvo glasovanje je prisutno još od popularizacije interneta. Mnoge države i organizacije su pokušale uvesti elektroničko glasovanje, neki su u tome i uspjeli te zadržali takav

sustav dok je većina došla do zaključka da rizici internet glasovanja nadmašuju koristi. Jedna država koja je u tome uspjela je Estonija. Tijekom lokalnih izbora 2005., Estonija je postala prva zemlja koja je imalo pravno obvezujuće opće izbore koristeći internet kao način glasovanja. Dvije godine kasnije u Estoniji se održavaju prvi nacionalni izbori s mogućnosti internet glasovanja u svijetu. Internet glasovanjem koristilo se 30 305 građana [2].

Glasovanje s pritiskom prsta iz udobnosti svog doma zvuči primamljivo, ali je vrijeme pokazalo da je teško izvedivo iz sigurnosnih razloga. Klasični načini glasovanja, iako po mnogima zastarjeli, imaju prednost u tome što je primjenjivano već stotinama godina, te su gotovo svi načini uplitanja u izbore već otkriveni, isprobani i preventirani. Internet je ujedno i najbolja i najgora stvar koja se mogla dogoditi glasovanju. Otvorena, globalna i anonimna priroda interneta kao medija kojim prolaze glasački listići otvara razne sigurnosne ranjivosti na cjelokupni sustav glasovanja koje prije nisu bili prisutne. Sigurnosni propust takvog glasovanja nije više lokalan kao što bi to bilo pri tradicionalnom glasovanju, gdje je uplitanje interesnih skupina u takve izbore prostorno i geografski ograničeno. To bi zahtijevalo da osoba, koja planira utjecati na izbore, fizički dođe do mjesta odvijanja izbora. Internet glasovanjem, takva sigurnosna ranjivost postaje globalna. Pojavljuje se potencijalna situacija gdje prosječni glasač ili prosječni organizator izbora postaje meta najboljih svjetskih hakera.

Napadi na elektronički sustav glasanja putem interneta, prema [3], su :

1. **Skalabilni**: Iskorištavanje sigurnosnih propusta mogu biti znatno pogubniji u domeni elektroničkog glasovanja. U situaciji gdje bi pojedinac pokušao promijeniti tijek glasanja na tipičnom glasovanju, postigao bi to na puno manjoj skali, na primjer, postigao bi to na samo jednoj izbornoj jedinici i time bi ugrozio relativno mal broj glasova u odnosu na cjelokupne izbore. Dok bi prilikom elektroničkog glasovanja jedna osobna mogla promijeniti rezultate cijelih izbora.

2. **Nevidljivi:** Napadač može promijeniti izborni ishod bez ikakvog značajnijeg rizika da izmjene budu uhvaćene. Takav napad postaje nemoguć za spriječiti ili ublažiti.

2.3. Blockchain glasovanje

Blockchain glasovanje najjednostavnije je objasniti kao podvrstu elektroničkog glasovanja s doprinosom digitalnih (kriptografskih) potpisa za otežavanje krivotvorenja glasova i korištenje hashiranja i distribuiranog konsenzusa za održavanje glasova na sigurnom.

On funkcionira na sljedeći način. Mreža računala zajednički pokreću programsku podršku kako bi se složili o redoslijedu i sastavu podataka. Takva, spomenuta, programska podrška je javna i dostupna svima. Korisnici dodaju nove podatke zajedno s digitalnim kriptografskim potpisima dok programska podrška nastoji validirati i povezati primljene podatke koristeći *hash* funkcije koje sprječavaju promjenu prijašnjih podataka. Prijašnji podaci tako ostaju trajni, nepromjenjivi i dostupni svim korisnicima. U kontekstu glasovanja, važno je istaknuti njegove sljedeće osobine [3]:

- **Konsenzus**. Konsenzus protokol predstavlja set pravila koje koriste čvorovi u mreži, na osnovu kojih se dolazi do zaključka što je najbolje za cijelu mrežu. Tako da, iako ne postoji središnje tijelo za provjeru valjanosti i verifikaciju, smatra se da je svaka transakcija u lancu potpuno osigurana i provjerena. Konsenzus ili protokol koji omogućava da se svako računalo može priključiti i sudjelovati u mreži se naziva *permissionless* protokol. Blockchain koji ga koristi oslovljavamo s javni i korišten je u većini trenutno najpopularnijih blockchainova (Bitcoin, Ethereum, Litecoin, Cardano, itd.). Pojava permissionless protocol-a natjerala je mnoge da ponovno sagledaju distribuirane baze podataka gdje je skup sudionika unaprijed određen i ograničen. Ovi protokoli poboljšavaju toleranciju grešaka pa čak mogu tolerirati i dio zlonamjernih čvorova. Takav protokol se naziva *permissioned* protokol, a blockchain koji ga koristi postaje privatan.

- **Autentikacija**. U sustavu blockchaina nema tradicionalnog korisničkog identiteta, jedino što korisnika predstavlja je privatni ključ kojim vrši digitalno potpisivanje. Korisnik je jedini odgovoran za upravljanje i pohranu svoga privatnog ključa. U slučaju da je privatni ključ ukraden ili izgubljen, korisnik gubi svoj „identitet" na blockchainu.

- **Pametni ugovori**. Pametni ugovor je računalni kod koji se automatski izvršava prilikom definiranih uvjetovanih događaja opisanih unutar ugovora. Na taj način je moguće izvršavanje puno kompleksnijih radnji na blockchainu od samih prijenosa vrijednosti s jedne adrese na drugu. Pomoću pametnih ugovora je moguće kreirati aplikacije poput tržnice, igara i raznih decentraliziranih financijskih aplikacija.

- **Tajnost transakcija**. Sve transakcije na blockchainu su javne, barem u većini slučajeva. To je jedna od ključnih značajki, da su sve transakcije transparentne i provjerljive. U privatnom blockchainu moguće je ograničiti pristup čitanju podataka, to može biti korisno

za ograničavanje curenja podataka ali oni bez pristupa ne mogu sudjelovati i provjeravati blockchain. Neki sustavi koriste „dokaz nultog znanja" za skrivanje pojedinosti o transakcijama, sudionike i iznos. Dokaz nultog znanja pouzdano pokazuje da je neka tvrdnja točna bez da otkriva zašto je ta izjava točna.

Najveći trag koji blockchain ostavlja na elektroničko glasovanje je koncept javno dostupne „glasačke kutije", gdje su sve transakcije odnosno glasački listići koji su trajno zapisani, nepromjenjivi i javno dostupni. No, mnogi stručnjaci tvrde da to nije bila najveća mana elektroničkog glasovanja, pa da je i taj dio nešto što se moglo riješiti tradicionalnom bazom podataka. Smatraju da problem leži u ranjivim elektroničkim uređajima i nesigurnom mediju preko kojeg se glasovi šalju [3].

Elektroničko glasovanje temeljeno na blockchainu obično izgleda ovako. Glasačko tijelo, koje vodi izbore i ima popis svih glasača, kreira par ključeva, javni i privatni, koji će predstavljati tog glasača. Uz glasačko pitanje i kandidate, definira se i vremenski okvir unutar kojeg će se moći glasovati. Glasači šalju svoje glasove potpisane njihovim privatnim ključem prije isteka vremenskog roka, nakon čega se izbori zatvaraju i glasovi prebrojavaju.

Prvi potencijalni problem koji se može pojaviti u ovom scenariju je da korisnik izgubi svoj par ključeva i tako izgubi svoj identitet. Ako korisnik izgubi svoj privatni ključ, on ne može više glasovati, a ako napadač dobije korisnikov privatni ključ, može neprimjetno glasovati. Blockchain u ovoj situaciji ne može nikako pomoći korisniku jer se cijeli sustav temelji na asimetričnoj kriptografiji - korištenju javnog i privatnog ključa za validaciju podataka. Taj se problem može riješiti tako da se glasačima ne daje njihov par ključeva direktno već da ih se pohrani u bazu podataka i veže za neki tradicionalniji sustav identifikacije ,na primjer korisnički račun s lozinkom. Iako se na taj način prvobitni problem riješio, ujedno se i poništilo sve što je taj par ključeva predstavljao, anonimnost i integritet glasača.

Kako se glasovi moraju poslati do određenog roka, glasač može biti uskraćen za glasovanje ako se njegov glas ne zaprimi do isteka vremena. Potencijalni napadač može privremeno uskratiti mrežne resurse pojedinom korisniku ili skupini DoS napadom (Denial-of-service attack [4]). Ovaj slučaj može ići i korak dalje ako se radi glasanju na javnom blockchainu. Javni blockchaini imaju ograničenu propusnost i zahtijevaju naknade za podnošenje transakcija. Kako promet raste, tako rastu i naknade za transakcije. Također, za vrijeme velikog prometa na blockchainu, transakcije se mogu i odgoditi. Interesna skupina koja želi sabotirati izbore može preplaviti blockchain

transakcijama do te mjere da glasačke transakcije postaju preskupe i/ili trajno odgođene. Napadač na taj način može spriječiti korisnike od glasovanja sve dok vremenski prozor za glasovanje ne istekne.

Glasovanje isto tako može postati i žrtva napada većine. Ako je cijeli blockchain ugrožen, ugroženo je i glasovanje. Organizacija odgovorna za napad većine u tom trenutku može manipulirati glasovanje sprječavanjem zapisa pojedinih transakcija. Sigurnost glasanja, naravno, ovisi i o sigurnom sklopovlju i programskoj podršci, na što sam blockchain ne utječe. Ako je uređaj za glasovanje korisnika ugroženo onda je zasigurno i njegov glas.

2.4. Voatz

Popularna implementacija blockchain glasovanja je aplikacija Voatz koja je korištena za neka od utjecajnijih nacionalna glasovanja u nekoliko američkih saveznih država. Svi glasovi dostavljeni na platformi Voatz generiraju papirnati glasački listić za tabeliranje, s potvrdom za glasače da potvrde svoj odabir. Od lipnja 2016. na platformi Voatz na više od 67 izbora glasovalo je više od 110.000 glasova. Nakon što korisnik preuzme aplikaciju Voatz, potvrđuje svoj telefonski broj, daje identifikacijski dokument s fotografijom, kao i „*selfie*". Prepoznavanje lica i popis glasača koriste se za provjeru identiteta i potvrdu podudarnosti između poslane slike i osobne iskaznice.

Postupak glasovanja preko aplikacije su naveli u četiri koraka [5]:

1. Glasač traži glasovanje u odsutnosti iz svoje jurisdikcije i pokazuje da bi želio „glasovati putem mobitela“

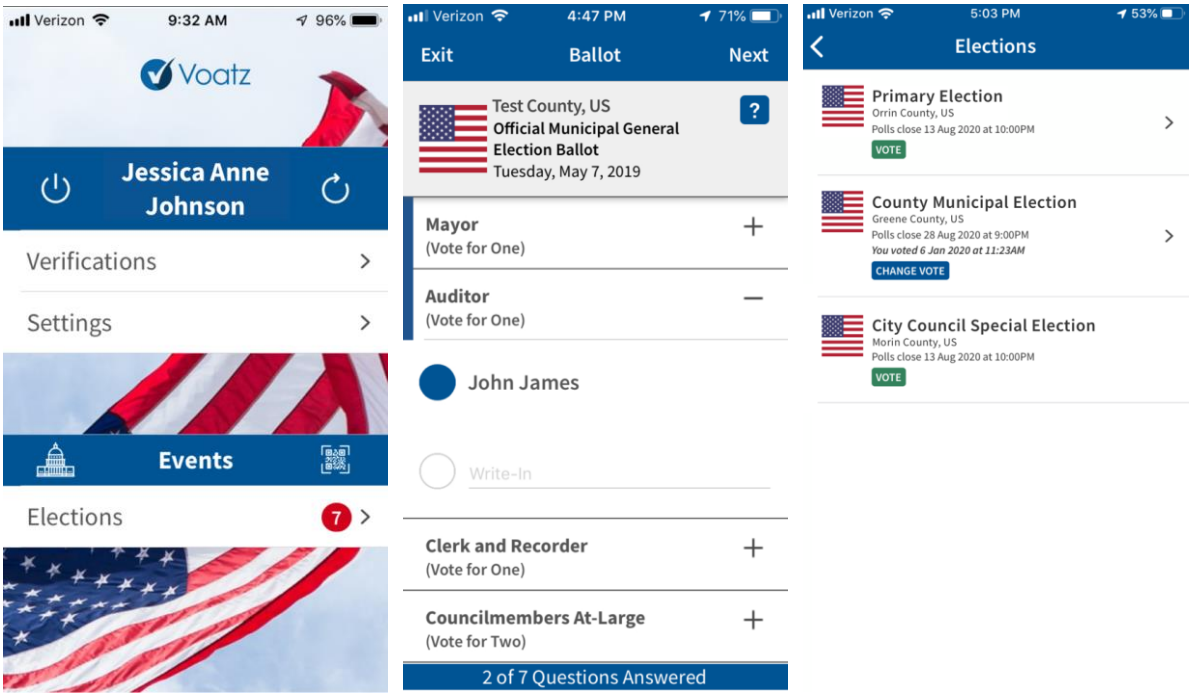
2. Glasač prima pozivnicu da preuzme aplikaciju Voatz i provjeri svoj identitet (obično skeniranjem službenog osobnog dokumenta s fotografijom).

3. Nakon što se identitet potvrdi, glasač glasa na pametnom telefonu, a u njegovoj jurisdikciji se proizvodi papirnati glasački listić. Glasač također dobiva glasački listić koji potvrđuje njihov odabir. Oba dokumenta digitalno su potpisana anonimnim ID-om radi očuvanja privatnosti.

4. Nakon izbora, revizija potvrđuje da tabelarni prikaz (papirnati glasački listići) odgovara izboru birača.

U službenom dokumentu o Voatz aplikaciji za glasovanje navode da su poboljšali elektroničko glasovanje u pogledu zaštite, identiteta i revizije [6]. No sigurnosni stručnjaci se nisu složili, navodeći kako nisu dovoljno otvoreni s implementacijom sustava glasovanja. Pruženi su površne

informacije o arhitekturi i korištenim kriptografskim postupcima što dosta ljudi smatra zabrinjavajućim [7]. Zaključuju da je najveći problem transparentnost i da javni izbori moraju biti bez vlasničke komponente, jer u suprotnom takav sustav postaje nepovjerljiv [8].



Sl. 2.1. *Prikaz Voatz aplikacije za glasovanje* [9]

2.5. Idejno rješenje aplikacije za glasovanje

Na primjeru Voatz aplikacije za blockchain glasovanje, koja je imala priliku održati značajne izbore u SAD-u, pokazano je da elektroničko glasovanje, bilo blockchain ili ne, još nije spremno za izbore najvišeg značaja. No to ne znači da takva aplikacija nema svoje tržište. Kada se govori o sigurnosnim propustima, ističe se da postoje scenariji u kojima se integritet glasovanja dovodi u pitanje. Da bi se takav scenarij dogodio, zlonamjerne skupine ili pojedinci moraju uložiti potrebno znanje, vještinu, vrijeme i novac. Usporedimo nacionalne izbore s glasovanjem unutar neke manje organizacije ili zajednice. S jednakim sigurnosnim manama, veća je vjerojatnost da će netko pokušati utjecati na izbore predsjednika države nego izbor za izbor predstavnika organizacije. Kako su veći ulozi tako su veći i resursi koje nekoga tko planira preokrenuti tijek izbora spreman uložiti. Sabotiranje glasovanja „nižeg profila“ zahtijeva jednake resurse za iskorištavanje izbornog sustava u svoju korist kao i nacionalni izbori. Elektroničko glasovanje, iako ima svoje sigurnosne probleme, može se i dalje bez brige koristiti za glasovanje unutar manjih zajednica, rezultati čijih

izbora neće utjecati na milijune ljudi. Iz tog razloga je takvo glasovanje znatno sigurnije, jer se uplitanje zlonamjernim osobama jednostavno ne isplati.

Nastavak rada će se usredotočiti na kreiranje upravo jedne takve aplikacije za glasovanje pomoću tehnologije blockchaina, koja će se koristiti za održavanje privatnih glasovanja unutar manjih vijeća, udruga i organizacija.

Funkcionalni zahtjevi koje će biti potrebno zadovoljiti su:

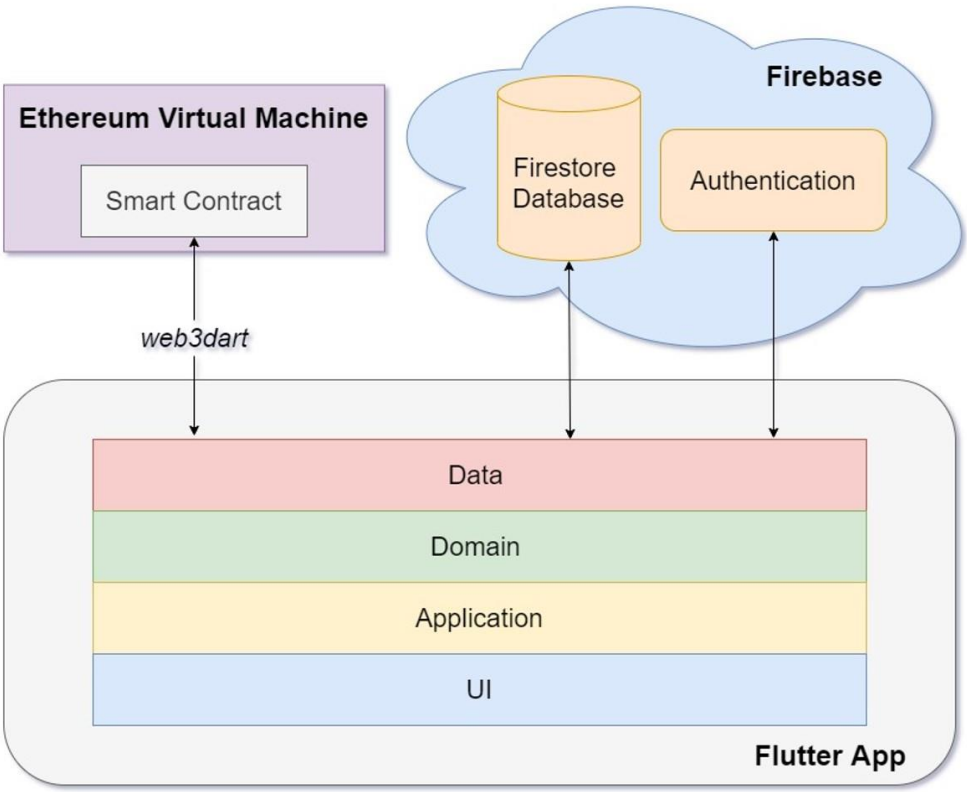
- Identifikacija korisnika preko korisničkog emaila i lozinke
- Svaki korisnik može pokrenuti nove izbore
- Svaki korisnik može pristupi izboru kreiranom od drugog korisnika ako posjeduje adresu izbora
- Svaki korisnik može pregledati svoju povijest glasovanja
- Korisnik koji je kreirao izbore treba imati mogućnost jednostavnog dijeljenja pristupa ostalim glasačima

3. PRIMIJENJENE TEHNOLOGIJE I ALATI

U nastavku je opisana arhitektura aplikacije za glasovanje pomoću blockchaina zajedno s opisom i ulogom svake od komponenata sustava.

3.1. Arhitektura

Arhitektura aplikacije za glasovanje je sljedeća, od vrha prema dnu. Na samom vrhu imamo ethereumov virtualni stroj i Firebase servis u oblaku. Ethereum virtualni stroj (engl. *Ethereum Virtual Machine*) ili skraćeno EVM je zadužen za izvođenje pametnih ugovora. Firebase servis u oblaku pruža uslugu autentikacije koja će se koristiti za identificiranje korisnika i pohranu njegovih podataka. Korisnik će biti pohranjen u Firestore bazi podataka zajedno sa kriptografskim elementima potrebnim za slanje i validaciju transakcija na Ethereum blockchainu. Mobilna aplikacija koja će komunicirati sa spomenutim servisima će biti napisana u Flutter programskom okviru za razvoj aplikacija koje se mogu izvoditi na više platformi. Naš fokus će biti na mobilnim platformama, iako se uz minimalne promjene programski kod može i prenamijeniti za desktop ili web uporabu. Svaka pojedina komponenta će biti detaljnije razjašnjena u sljedećim potpoglavljima.



Sl. 3.1. *Arhitektura aplikacija za glasovanje*

3.2. Ethereum

U aplikaciji korišten Ethereum blockchain kao javna, decentralizirana, nepromjenjiva baza podataka. Iako nije najbrži, najsigurniji niti najpovoljniji, zasigurno je najpristupačniji za razvoj decentraliziranih aplikacija. Pokrenut je 2015., što ga čini najzrelijim programabilnim blockchainom. Od tada je imao priliku skupiti najveću zajednicu razvojnih programera koji svakodnevno razvijaju i nadograđuju alate i sučelja koji olakšavaju razvoj decentraliziranih aplikacija na različitim platformama. Za razumijevanje tehnologije i uloge u aplikaciji za glasovanje vrlo su bitna dva pojma [10]:

- **Ethereum virtualni stroj** je runtime okruženje. Što znači da omogućuje pokretanje koda pametnog ugovora prevedenog (engl. *compiled*) u prijenosni kod (engl. *bytecode*). EVM je instaliran na svakom čvoru u mreži i radi kao srednji sloj između pametnog ugovora i operacijskog sustava.

- **Pametni ugovori** predstavljaju programski kod koji se izvodi povrh blockchaina. Ako je blockchain podatkovni sloj onda bi pametni ugovori predstavljali sloj poslovne logike. Sadrže set instrukcija koji se izvodi nad podacima zapisanim u lancu blokova. Razvojni programeri koriste pametne ugovore kako bi proširili funkcionalnost blockchaina te na taj način pružili složeniju uslugu krajnjem korisniku. Dva najčešće korištena jezika za razvoj pametnih ugovora su Solidity i Vyper.

Sl. 3.2. *Grafički prikaz izvođenja pametnog ugovora*

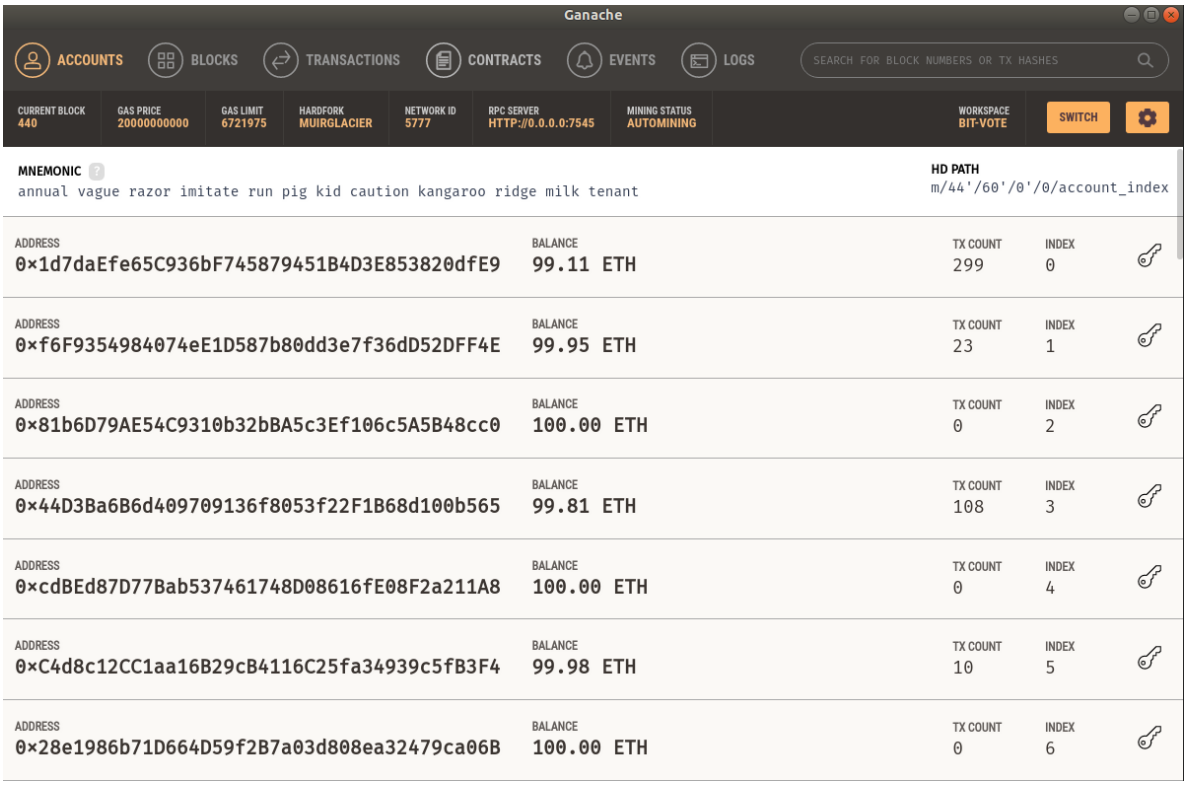
3.2.1. Solidity

Solidity je objektno orijentirani jezik visoke razine namijenjen za razvijanje pametnih ugovora na EVM-u. Po sintaksi je nalik neizbježnom JavaScriptu što ga čini vrlo pristupačnim [11]. Neke od značajki :

- složeni tipovi podataka
- tip varijable je poznat za vrijeme izvođenja
- podržava nasljeđivanje
- podržava biblioteke

3.2.2. Truffle & Ganache

Truffle je razvojno okruženje i okvir za testiranje. Pruža usluge programskog prevođenja, implementacija i povezivanja pametnih ugovora. Truffle u kombinaciji s Ganacheom pruža produktivno okruženje za razvoj decentraliziranih aplikacija. Kako je Truffle zadužen za pametne ugovore, Ganache omogućava stvaranje vlastitog privatnog blockchaina na kojemu možemo testirati pametne ugovore bez brige o propusnosti i naknadama blockchaina. Automatski postavlja nekoliko ethereum adresa zajedno s privatnim ključevima i simuliranim *ether*-om. [12]



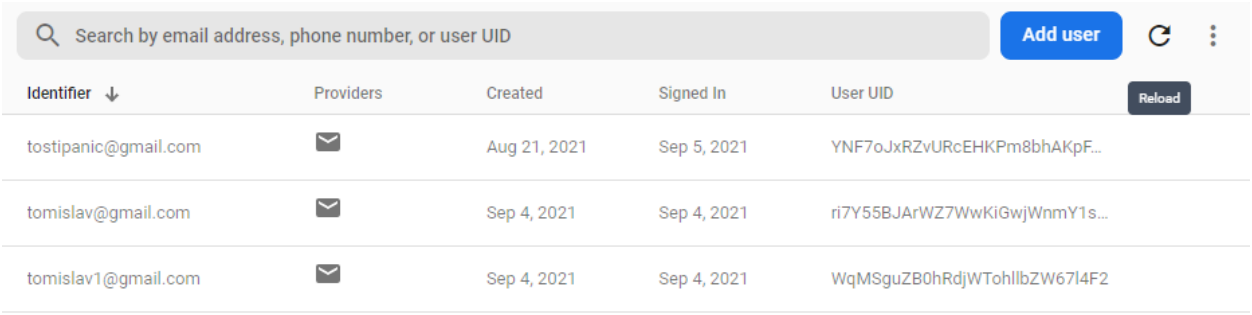
Sl. 3.3. *Ganache korisničko sučelje*

3.1. Firebase

Firebase je servis koji pruža usluge pozadinske programske podrške (engl. *Backend-as-a-Service*). Podržavan je od strane Googlea, a nudi niz usluga i alata, uključujući praćenje analitike, izvještavanje, slanje obavijest, itd. [13]. U aplikaciji za glasovanje pomoću tehnologije blockchaina, koristit ćemo Firebase za autentikaciju korisnika i pohranu podataka u Firestore bazu podataka.

3.1.1. Autentikacija

Firebase autentikacija podržava provjeru autentičnosti pomoću lozinki, telefonskih brojeva, popularnih davatelja identiteta poput Googlea, Facebooka i Twittera itd. U aplikaciji se koristi autentikacija pomoću korisničkog imena i lozinke.

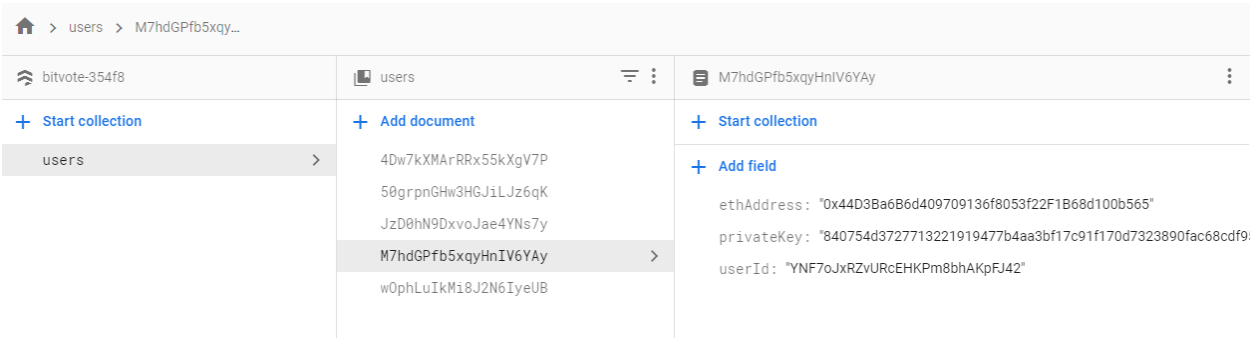


Sl. 3.4. *Firebase autentikacija*

3.1.2. Firestore baza podataka

Firestore je NoSQL baza podataka orijentirana na dokumente. Za razliku od SQL baze podataka, nema tablica ili redaka. Umjesto toga, podaci se pohranjuju u dokumente koji su organizirani u zbirke. Zbirke možemo zamisliti kao ekvivalent tablicama u SQL bazama podataka, dok bi dokumenti predstavljali retke. Svi dokumenti moraju biti pohranjeni u zbirkama.

Prilikom registracije na aplikaciju, ID korisnika će se pohraniti u bazu podataka pri čemu će se korisniku pridodati privatni ključ i ethereum adresa koja će ga predstavljati na blockchainu.



Sl. 3.5. *Firestore baza podataka*

3.2. Flutter

Flutter predstavlja set alata za razvoj korisničkog sučelja. Otvorenog je koda, a pisan je u programskom jeziku Dart. Glavna značajka Flutter-a je mogućnost razvoja višeplatformskih aplikacija. Sa samo jednom kodnom bazom možemo razvijati aplikacije za različite platforme. Trenutno podržane platforme uključuju: mobilne platforme(iOS, Android), web i desktop(Windows, Linux, macOS) [14].

3.2.1. Domain Driven Design

Razvoj Flutter mobilne aplikacije je baziran na arhitekturalnom oblikovnom obrascu pod nazivom *Domain Driven Design* (krat. DDD). Zadaća oblikovnih obrazaca je rješavanje učestalih problema u različitim situacijama. Primjerice, zadaća DDD oblikovnog obrasca je pružiti fleksibilnost u razvoju omogućujući redovito i kontinuirano održavanje, mijenjanje i poboljšanje sustava [15]. Glavna kritika ovoga oblikovnog obrasca je količina potrebne dodatne implementacije u obliku enkapsulacije koda kako bi se model održao čistim i održivim. DDD razdvaja programski kod na sljedeće slojeve:

Korisničko sučelje – odgovorno za prikazivanje informacija i interakciju s korisnikom

Sloj aplikacije – definira poslove i usmjerava objekte domene na rješavanje problema

Sloj domene – predstavlja koncepte, pravila i poslovnu logiku.

Sloj podataka – sloj koji međudjeluje s vanjskim sustavima razmjenjujući podatke

3.2.2. Web3

Pojam web3 predstavlja ideju o trećoj generaciji interneta, pri čemu prva iteracija označava pojavu statičkih web stranica dok web2.0 donosi interaktivnost i društveni aspekt. Iako se web 3.0 još nije dogodio, mnogi pretpostavljaju da će iduća evolucija interneta doći u obliku decentralizacije. Web3 je ujedno i naziv za kolekciju biblioteka za interakciju s lokalnim ili udaljenim ethereum čvorom koristeći HTTP, IPC ili WebSocket. U aplikaciji je korištena Web3dart biblioteka [16], inačica biblioteke prilagođena za Dart programski jezik. Web3dart je korišten za RPC spajanje na ethereum čvor, pozivanje funkcija i slušanje evenata pametnog ugovora.

4. PROGRAMSKO RJEŠENJE APLIKACIJE ZA GLASOVANJE

U ovom je poglavlju prikazano programsko rješenje aplikacije za glasovanje podijeljeno na slojeve, od Ethereum pametnih ugovora pisanih u Solidity programskom jeziku pa sve do korisničkog sučelja višeplatformske mobilne aplikacije.

4.1. Pametni ugovor

Za uspješnu komunikaciju s ethereum blockchainom potrebno je definirati vrstu i strukturu podataka koje će se pohranjivati na blockchainu i funkcije kojima ćemo utjecati na te iste podatke.

Struktura podataka je prikazana slikom 4.1.

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

contract Elections {
    struct Candidate {
        uint256 id;
        string name;
    }

    struct Voter {
        address voterAddress;
        bool isRegistered;
        bool hasVoted;
        uint256 voteTowards;
    }

    enum State {
        CREATED,
        ONGOING,
        CONCLUDED
    }

    struct BallotBox {
        uint256 id;
        address admin;
        string description;
        uint256 candidate_count;
        mapping(uint256 => Candidate) candidates;
        uint256 voter_count;
        mapping(address => Voter) voters;
        uint256 startingTime;
        uint256 duration;
        uint256 endTime;
        mapping(uint256 => uint256) voteCount;
        State electionState;
    }

    mapping(uint256 => BallotBox) private boxes;
    uint256 ballotCount;
```

Sl. 4.1. *Struktura podatka pametnog ugovora*

Unutar Solidity-a, ključna riječ struci predstavlja novi tip podataka s 2 ili više člana. U ovom slučaju kreirana su četiri nova tipa podataka: *Candidate, Voter, State* i *BallotBox*.

Candidate je izborna opcija, *Voter* je glasač na izborima koji posjeduje ethereum adresu, identifikaciju izborne opcije i dvije boolean vrijednosti po kojima validiramo glasačev status pri glasanju, a State tipom podataka pratimo tijek glasačkih izbora. *BallotBox* je glavni model podataka koji objedinjuje ostale već spomenute modele u jednu cjelinu. Jedan *BallotBox* (engl. Glasačka kutija) predstavlja jedne glasačke izbore pa tako sadrži adresu pokretača izbora, izborno pitanje, izborne opcije, glasače, trajanje izbora, status i rezultate izbora. Tipovi podataka korištenih su većinom string, boolean i 256-bitni pozitivni cijeli broj. Korištena je i ključna riječ *mapping* koja kreira ključ-vrijednost strukturu podataka između dva tipa podataka navedenih unutar zagrade.

Prije samih funkcija pametnog ugovora, važno je spomenuti ključne riječi *event* i *modifier*. *Event* je zadužen za emitiranje poruka. Kako izvršavanje blockchain transakcije ovisi o protočnosti, odnosno prometu blockchaina na kojemu se izvodi, nije moguće očekivati povratnu informaciju o promjeni na blockchainu. Za razliku od tradicionalnog HTTP POST zahtjeva, gdje se nakon poslanih ulaznih podataka očekuje odgovor o promjeni, na blockchainu to nije moguće. It tog je razloga omogućeno kreiranje *event* funkcija, čija je zadaća obavještavanje o događajima pri izvođenju pametnog ugovora.

```
118
119     event BallotBoxCreated(address sender, uint256 boxId, string desc, State electionState);
120     event ElectionStart(address sender, uint256 boxId, State electionState, uint256 endTime);
121     event ElectionEnd(address sender, uint256 boxId, State electionState);
122     event AddedAVoter(address sender, uint256 boxId);
123     event AddedACandidate(address sender, uint256 boxId, uint256 candidateID, string candidateName);
124     event VotedSuccessfully(address sender, uint256 boxId, uint256 candidateId);
```

Sl. 4.2. *Događaji pametnog ugovora*

```
39      modifier checkAdmin(uint256 _boxId, address _owner) {
40          require(
41              boxes[_boxId].admin == _owner,
42              "Only the election admin has access to this function."
43          );
44          _;
45      }
46
47      modifier checkIfCreated(uint256 _boxId) {
48          require(
49              boxes[_boxId].electionState == State.CREATED,
50              "The election is either ongoing or has concluded."
51          );
52          _;
53      }
```

Sl. 4.3. *Validacija pametnog ugovora*

Modifier se može opisati kao pomoćna funkcija kojom provjeravamo ulazne podatke i stanje blockchaina prije samog izvršavanja funkcija. Unutar *modifier* funkcije se definira uvjet koji želimo provjeriti i poruka koja će se prikazati ako uvjet nije ispunjen. Na slici 4.3 prikazana je definirana modifier funkcija.

Kreirane Solidity funkcije za elektroničko glasovanje se mogu podijeliti u dvije grupe: funkcije koje mijenjaju stanje na blockchainu i funkcije koje isključivo čitaju podatke s blockchaina. Te dvije grupe se u kodu razlikuju po ključnoj riječi *view* koja označava funkciju koja neće mijenjati varijable niti emitirati događaje. Na slici 4.4 je prikazana funkcija koja ne mijenja stanje blockchaina dok su na slici 4.5 prikazane funkcije koje mijenjaju.

```
258     function showBallotBox(uint256 _boxId)
259         public
260         view
261         returns (
262             uint256 boxId,
263             address admin,
264             string memory state,
265             uint256 end,
266             string memory desc,
267             string memory candidates,
268             uint256[] memory
269         )
270     {
271         uint256[] memory candidateVotes = new uint256[](
272             boxes[_boxId].candidate_count
273         );
274         string memory candidateNames = "";
275         for (uint256 i = 0; i < boxes[_boxId].candidate_count; i++) {
276             candidateVotes[i] = boxes[_boxId].voteCount[i + 1];
277             candidateNames = append(
278                 candidateNames,
279                 boxes[_boxId].candidates[i + 1].name
280             );
281         }
282 
283         return (
284             _boxId,
285             boxes[_boxId].admin,
286             checkState(_boxId),
287             boxes[_boxId].endTime,
288             boxes[_boxId].description,
289             candidateNames,
290             candidateVotes
291         );
292     }
293 }
```

Sl. 4.4. *Funkcija pametnog ugovora za čitanje podataka o glasačkoj kutiji*

```
186     function addCandidate(uint256 _boxId, string memory _name)
187         public
188         checkIfCreated(_boxId)
189         checkAdmin(_boxId, msg.sender)
190     {
191         boxes[_boxId].candidate_count++;
192         boxes[_boxId].candidates[boxes[_boxId].candidate_count].id = boxes[
193             _boxId
194         ].candidate_count;
195         boxes[_boxId].candidates[boxes[_boxId].candidate_count].name = _name;
196         boxes[_boxId].voteCount[boxes[_boxId].candidate_count] = 0;
197         emit AddedACandidate(
198             msg.sender,
199             _boxId,
200             boxes[_boxId].candidate_count,
201             _name
202         );
203     }
204 
205     function addVoter(uint256 _boxId)
206         public
207         checkNotRegistered(_boxId, msg.sender)
208         checkNotComplete(_boxId)
209     {
210         boxes[_boxId].voter_count++;
211         boxes[_boxId].voters[msg.sender].voterAddress = msg.sender;
212         boxes[_boxId].voters[msg.sender].isRegistered = true;
213         emit AddedAVoter(msg.sender, _boxId);
214     }
215 
216     function vote(uint256 _boxId, uint256 _candidateId)
217         public
218         checkIfOngoing(_boxId)
219         checkIfVoterValid(_boxId, msg.sender)
220         checkIfCandidateValid(_boxId, _candidateId)
221     {
222         boxes[_boxId].voters[msg.sender].hasVoted = true;
223         boxes[_boxId].voters[msg.sender].voteTowards = _candidateId;
224         boxes[_boxId].voteCount[_candidateId] += 1;
225         emit VotedSuccessfully(msg.sender, _boxId, _candidateId);
226     }
```

Sl. 4.5. *Funkcije pametnog ugovora koje mijenjaju stanje blockchaina*

Kako bi vanjska aplikacija bila u mogućnosti razmjenjivati podatke s EVM-om potrebno je generirati ABI. ABI(*application binary interface*) stoji za binarno sučelje aplikacije koje sadrži sve podatke o definiranim funkcijama unutar pametnog ugovora. To znači da vanjska aplikacija mora biti upoznata s ulaznim i izlaznim parametrima funkcija kako mi mogla iščitavati podatke iz transakcija.

4.2. Sloj podataka

Sloj podataka je prvi sloj unutar kreirane aplikacije za glasovanje koji jedini direktno komunicira s vanjskim servisima. U ovom slučaju, to je Firebase servis za autentikaciju, Firebase baza podataka i ethereum blockchain.

4.2.1. Firebase Autentikacija

Za potrebe korištenja Firebase autentikacije, kreirana je klasa koja enkapsulira instancu Firebase autentikacije i metode za prijavu i registraciju korisnika. Funkcija za registraciju sa slike 4.6 prima email adresu i lozinku u obliku tipa podataka koji su posebno kreirani za njihovu ulogu. Tako generirani objekti prolaze kroz dodatni sloj validacije prilikom kreiranja što ih čini pouzdanijima. Nešto više o tome će biti opisano u potpoglavlju sloja domene. Funkcija za registraciju daje kao rezultat asinkronu vrijednost koja za vrijeme izvođenja može biti ili iznimka ili *Unit* (povratna vrijednost koja ne pohranjuje nikakve informacije). Iz primjera vidimo da funkcija može rezultirati iznimkom neočekivane vrijednost, iznimkom već korištene email adrese i iznimkom greške na poslužitelju.

```
class FirebaseAuthentication implements IFirebaseAuth {
  FirebaseAuthentication(this._firebaseAuth);

  final FirebaseAuth _firebaseAuth;

  @override
  Future<Either<AuthFailures, Unit>> registerWithEmailAndPassword(
      {required EmailAddress? emailAddress,
      required Password? password}) async {
    final emailAddressString = emailAddress!.valueObject!
        .fold((l) => throw UnExpectedValueError(l), id);
    final passwordString =
        password!.valueObject!.fold((l) => throw UnExpectedValueError(l), id);
    try {
      await _firebaseAuth.createUserWithEmailAndPassword(
          email: emailAddressString, password: passwordString);
      return right(unit);
    } on FirebaseAuthException catch (e) {
      if (e.code == 'email-already-in-use') {
        return left(const AuthFailures.emailAlreadyInUse());
      } else {
        return left(const AuthFailures.serverError());
      }
    }
  }
}
```

Sl. 4.6. *Prikaz funkcije za registraciju pomoću Firebase autentikacije*

4.2.2. Firebase Firestore

Firestore baza podataka se u aplikaciji koristi kao način pohrane podataka izvan blockchaina. To znači da se svi podaci vezani za identifikaciju na blockchain spremaju na Firestore bazu podataka. Identifikacija dobivena prilikom registracije i prijave korisnika se veže za već postojeću ethereum adresu i privatni ključ, te se kao takva pohranjuje u bazu. Uz identifikaciju korisnika, u bazu se pohranjuju i adrese prijašnjih glasovanja korisnika na blockchainu kako bi se naknadno mogla dohvaćati povijest korisničkih glasovanja. Interakcija s Firestore servisom djeluje jednako kao i s Firebase Autentikacijom. Kreirana je klasa koja posreduje između servisa i aplikacije u kojoj su definirane metode za radnje na bazi. Jedna od tih metoda za dohvaćanje podataka o korisniku je prikazana na slici 4.7.

```
@override
Future<Either<FirestoreFailures, UserData>> readUserData() async {
  try {
    return _firebaseFirestore
        .collection("users")
        .where("userId", isEqualTo: FirebaseAuth.instance.currentUser!.uid)
        .limit(1)
        .get()
        .then((value) {
      final document = value.docs[0];
      var documentData = document.data();
      document.reference.update(documentData);
      return right(
        UserData(
          address: Address(
            address: documentData["ethAddress"],
          ),  // Address
          privateKey: PrivateKey(
            privateKey: documentData["privateKey"],
          ),  // PrivateKey
        ),  // UserData
      );
    });
  } catch (e) {
    if (e is FirebaseException && e.message!.contains('PERMISSION_DENIED')) {
      return left(const FirestoreFailures.unauthorized());
    } else {
      return left(const FirestoreFailures.serverError());
    }
  }
}
```

Sl. 4.7. *Prikaz funkcije za čitanje podataka o korisniku*

4.2.3. Web3

Aplikacija za glasovanje komunicira s blockchainom preko Web3Service klase koje je napisana uz pomoć web3dart biblioteke. Kod za instanciranje takve klase možemo vidjeti na slici 4.8 Za instanciranje je potrebno imati ethereum adresu i privatni ključ trenutnog korisnika aplikacije, također unutar klase je potrebno unijeti i adresu RPC poslužitelja, adresu pametnog ugovora na ethereum blockchainu i ABI JSON dokument.

```
class Web3Service implements IWeb3Service {
  static final Web3Service _inst = Web3Service._internal();

  Web3Service._internal();

  factory Web3Service(String privateKey, String ethAddressHex) {
    if (_inst._privateKey != privateKey) {
      _inst._client = Client();
      _inst._web3Client = Web3Client(RPC_SERVER, _inst._client);
      _inst.ethAddress = EthereumAddress.fromHex(ethAddressHex);
      _inst._privateKey = privateKey;
    }
    return _inst;
  }

  factory Web3Service.instance() {
    return _inst;
  }

  final _logger = Logger();

  String _privateKey = "";
  late EthereumAddress ethAddress;
  late Client _client;
  late Web3Client _web3Client;

  Future<DeployedContract> loadContract() async {
    String abi = await rootBundle.loadString(ABI_PATH);
    String contractAddress = CONTRACT_ADDRESS;

    final contract = DeployedContract(
      ContractAbi.fromJson(abi, "Elections"),
      EthereumAddress.fromHex(contractAddress),
    );  // DeployedContract
    return contract;
  }
```

Sl. 4.8. *Prikaz inicijalizacije Web3Service klase*

U Web3Service klasi je definirana i privatna metoda *_sendTransaction* (Sl.4.9) koja se poziva pri obavljanju svake transakcije jer kao takva sadrži sve potrebne korake za obavljanje pri svakoj transakciji.

```
Future<String> _sendTransaction(
    String functionName, List<dynamic> args) async {
  DeployedContract _contract = await loadContract();
  DeployedContract contract = _contract;
  final ethFunction = contract.function(functionName);
  final _credentials = EthPrivateKey.fromHex(_privateKey);

  final result = await _web3Client.sendTransaction(
    _credentials,
    Transaction.callContract(
      contract: contract,
      function: ethFunction,
      parameters: args,
    ),
  );
  return result;
}
```

Sl. 4.9. *Prikaz funkcije za slanje transakcija*

Jedna od funkcija s pametnog ugovora koju pozivamo iz ove klase je *createBallotBox* (Sl 4.10) koja za argumente prima izborno pitanje a kao rezultat ne vraća ništa. Kako je i u prijašnjem poglavlju spomenuto, funkcije koje mijenjaju stanje blockchaina nemaju povratnu vrijednost. Razlog tome je što web3 biblioteka ne može znati kada će se transakcija pridružiti bloku. Postojeća alternativa je kreiranje događaja. Događaj se može kreirati i pozivati unutar funkcija pametnog ugovora. Takvi događaji se tada mogu slušati unutar aplikacije, tako da pri svakoj pojavi događaja naša aplikacija primi informacije o izvršenoj transakciji. Jedan takav događaj je i *votedSuccessfullyEvent* prikazan na slici 4.11, koji se izvodi pri izvršavanju transakcije funkcije *vote*.

```
Future<Either<BlockchainFailures, Unit>> createElection(
    {required Topic? topic}) async {
  final topicValue =
      topic!.valueObject!.fold((l) => throw UnExpectedValueError(l), id);

  try {
    var response = await _sendTransaction("createBallotBox", [topicValue]);
    print(response);
    return right(unit);
  } catch (e) {
    print(e.toString());
    return left(const BlockchainFailures.transactionFailed());
  }
}
```

Sl. 4.10. *Prikaz funkcije za kreiranje nove glasačke kutije*

```
void votedSuccessfullyEvent(
    void onVotedSuccessfully(
        String sender, BigInt ballotBox, BigInt candidateId)) async {
  DeployedContract _contract = await loadContract();
  final votedSuccessfully = _contract.event('VotedSuccessfully');

  _web3Client
      .events(
    FilterOptions.events(
      contract: _contract,
      event: votedSuccessfully,
    ),
  )
      .listen((event) {
    final decoded =
        votedSuccessfully.decodeResults(event.topics!, event.data!);
    final sender = decoded[0] as EthereumAddress;
    if (ethAddress == sender) {
      final ballotBoxId = decoded[1] as BigInt;
      final candidateId = decoded[2] as BigInt;

      onVotedSuccessfully(sender.hex, ballotBoxId, candidateId);
    }
  });
}
```

Sl. 4.11. *Prikaz funkcije koja se izvodi pojavom događaja uspješnog glasanja*

4.3. Sloj domene

Sloj domene je sloj koji je nastanjen entitetima, atributima, pravilima i procesima podijeljenim u više grupa koje su određene problemima koje rješavaju. Takvu grupu nazivamo domena. Različiti problemi zahtijevaju različite domene. U aplikaciji za glasovanje je kreirano više različitih domena, gdje svaka rješava jedan jasno definirani problem. Pa su tako kreirane domene autentikacije, kreiranja novih izbora, glasanja, itd..

Na primjer, domena autentikacije je definirana na sljedeći način. Prvo je potrebno definirati objekte koji se koriste u domeni autentikacije, korisnički email i lozinku. Na slici 4.12 je priložen kod koji opisuje upravo to. Klase *EmailAddress* i *Password* nasljeđuju klasu *ValueObject* čija je namjena da bude temeljni objekt domene koja sadrži metode koje su potrebne svim ostalim entitetima domene.

```dart
class EmailAddress extends ValueObject<String> {
  factory EmailAddress({String? email}) {
    return EmailAddress._(validateEmailAddress(email: email));
  }

  const EmailAddress._(this.valueObject);

  @override
  final Either<ValueFailures<String>, String>? valueObject;
}

class Password extends ValueObject<String> {
  factory Password({String? password}) {
    return Password._(validatePassword(password: password));
  }

  const Password._(this.valueObject);

  @override
  final Either<ValueFailures<String>, String>? valueObject;
}
```

Sl. 4.12. *Prikaz EmailAddress i Password entiteta*

Prilikom definiranja entiteta domene, nužno je dodati validatore koji će pomoći da se prilikom izvođenja aplikacija služi samo ispravnim entitetima. Za domenu autentikacije se koristi validator email adresa (Sl. 4.11) i validator lozinki (Sl. 4.12).

```dart
Either<ValueFailures<String>, String> validateEmailAddress({
  required String? email,
}) {
  final emailRegex = RegExp(
      r'^[a-zA-Z0-9.a-zA-Z0-9.!#$%&"*+-/=?^_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-Z]+');

  if (emailRegex.hasMatch(email!)) {
    return right(email);
  } else {
    return left(
      ValueFailures.invalidEmail(failedValue: email),
    );
  }
}
```

Sl. 4.13. *Prikaz validatora email adresa*

```dart
Either<ValueFailures<String>, String> validatePassword({
  required String? password,
}) {
  final hasMinLength = password!.length > 6;
  final hasUppercase = password.contains(RegExp('[A-Z]'));
  final hasDigits = password.contains(RegExp('[0-9]'));
  final hasSpecialCharacters =
      password.contains(RegExp(r'[!@#$%^&*(),.?":{}|<>]'));
  if (!hasMinLength) {
    return left(
      ValueFailures.shortPassword(failedValue: password),
    );
  } else if (!hasUppercase) {
    return left(
      ValueFailures.noUpperCase(failedValue: password),
    );
  } else if (!hasDigits) {
    return left(
      ValueFailures.noNumber(failedValue: password),
    );
  } else if (!hasSpecialCharacters) {
    return left(
      ValueFailures.noSpecialSymbol(failedValue: password),
    );
  } else {
    return right(password);
  }
}
```

Sl. 4.14. *Prikaz validatora lozinki*

U slučaju da validator pronađe neispravnu vrijednost entiteta, vraća se poseban *failure* objekt koji se u kodu može dalje koristiti za bolje upravljanje greškama. *Failure* objekt može biti specifičan za neku domenu(Sl.4.15) ili za neku vrijednost (Sl.4.16).

```dart
@freezed
class AuthFailures with _$AuthFailures {
  const factory AuthFailures.serverError() = ServerError;

  const factory AuthFailures.emailAlreadyInUse() = EmailAlreadyInUse;

  const factory AuthFailures.invalidEmailAndPasswordCombination() =
  InavalidEmailAndPasswordCombination;
}
```

Sl. 4.15. *Prikaz klase AuthFailures*

```dart
@freezed
class ValueFailures<T> with _$ValueFailures<T> {
  const factory ValueFailures.invalidEmail({required String? failedValue}) =
      InvalidEmail<T>;

  const factory ValueFailures.shortPassword({required String? failedValue}) =
      ShortPassword<T>;

  const factory ValueFailures.longString({required String? failedValue}) =
      LongString<T>;

  const factory ValueFailures.longList({required List<dynamic>? failedValue}) =
  LongList<T>;

  const factory ValueFailures.noSpecialSymbol({required String? failedValue}) =
      NoSpecialSymbol<T>;

  const factory ValueFailures.noUpperCase({required String? failedValue}) =
      NoUpperCase<T>;

  const factory ValueFailures.noNumber({required String? failedValue}) =
      NoNumber<T>;

  const factory ValueFailures.negativeId({required BigInt? failedValue}) =
      NegativeId<T>;

  const factory ValueFailures.invalidPrivateKey({required String? failedValue}) =
  InvalidPrivateKey<T>;

  const factory ValueFailures.invalidEthAddress(
      {required String? failedValue}) = InvalidAddress<T>;
}
```

Sl. 4.16. *Prikaz klase ValueFailures*

4.4. Sloj aplikacije

Sloj aplikacije je sloj koji ima zadaću usmjeravanja objekata domene prema rješavanju problema. To je mjesto gdje se implementiraju svi slučajevi upotrebe koji ovise o korisničkom sučelju. U aplikaciji za glasovanje, sloj aplikacije je ostvaren preko interakcije između događaja i stanja.

4.4.1. Stanje

Stanje predstavlja informacije na osnovu kojih se korisničko sučelje mijenja. Na slici 4.17 je predočena klasa stanja za zaslon kreiranja glasačke kutije. Unutar klase su sve vrijednosti koje su potrebne korisničkom sučelju da na osnovu njih napravi promjenu na korisnikovom zaslonu. Stanje se može promijeniti s pojavom „događaja“ kojega najčešće zaziva korisnik interakcijom sa zaslonom aplikacije.

```
@freezed
class CreateBallotStates with _$CreateBallotStates {
  const factory CreateBallotStates({
    required Id id,
    required Address adminAddress,
    required Topic topic,
    required List<Candidate> candidates,
    required List<Voter> voters,
    required ElectionState electionState,
    required BigInt duration,
    required bool isSubmitting,
    required bool showError,
    required String status,
    required int selectedCandidate,
    required Option<Either<BlockchainFailures, Unit>>
        transactionFailureOrSuccess,
  }) = _CreateBallotStates;

  factory CreateBallotStates.initial() => CreateBallotStates(
        id: Id(id: BigInt.zero),
        adminAddress: Address(address: ''),
        topic: Topic(topic: ''),
        candidates: List<Candidate>.of([]),
        voters: List<Voter>.of([]),
        electionState: ElectionState.NOT_CREATED,
        duration: BigInt.zero,
        status: "",
        selectedCandidate: 0,
        isSubmitting: false,
        showError: false,
        transactionFailureOrSuccess: none(),
      );  // CreateBallotStates
}
```

Sl. 4.17. *Prikaz klase stanja za kreiranje nove glasačke kutije*

4.4.2. Događaj

Događaj je pojava unutar aplikacije koja izaziva nekakvu reakciju. Ta reakcija je najčešće promjena stanja unutar aplikacije. Događaj se može pokrenuti s bilo kojeg mjesta u programskom kodu, primjerice na korisničkom sučelju. Može se pojaviti i prilikom izvršavanja zadataka, promjenom u statusu mrežnog povezivanja, promjena očitanja senzora itd. Slika 4.18 sadrži prikaz događaja definiranih za zaslon kreiranja glasačke kutije. Na primjer, možemo izazvati događaj prilikom promjene trajanja izbora na zaslonu, a može se izazvati događaj i prilikom uspješne transakcije za kreiranje glasačke kutije na ethereum blockchainu.

```
@freezed
class CreateBallotEvents with _$CreateBallotEvents {
  const factory CreateBallotEvents.onEditTopicChanged({
    required String? topic,
  }) = OnEditTopicChanged;

  const factory CreateBallotEvents.onEditCandidateName({
    required BigInt? candidateId,
    required String? candidateName,
  }) = OnEditCandidateName;

  const factory CreateBallotEvents.onEditDuration({
    required BigInt? duration,
  }) = OnEditDuration;

  const factory CreateBallotEvents.onEditCandidateAdded() = OnEditCandidateAdded;

  const factory CreateBallotEvents.onRemoveCandidate({
    required BigInt? candidateId,
  }) = OnRemoveCandidate;

  const factory CreateBallotEvents.ballotBoxCreated({
    required String? sender,
    required BigInt? ballotBoxId,
    required String? topic,
    required String? electionState,
  }) = BallotBoxCreated;

  const factory CreateBallotEvents.ballotBoxStarted({
    required String? sender,
    required BigInt? ballotBoxId,
    required BigInt? electionState,
    required BigInt? endTime,
  }) = BallotBoxStarted;

  const factory CreateBallotEvents.createBallotBox() = CreateBallotBox;
}
```

Sl. 4.18. *Prikaz klase događaja za kreiranje nove glasačke kutije*

4.4.3. Upravljanje slučajevima

Uz stanja i događaje, postoji i klasa koja spaja pojavu događaja s promjenom stanja. Drugim riječima, definira se što će se dogoditi sa stanjem ako dođe do određenog događaja. U domeni kreiranja glasačke kutije, takvo ponašanje je definirano unutar *CreateBallotStateController* klase prikazane na slici 4.19. Primjerice, prilikom pritiska na dugme za dodavanje novog kandidata na korisničkom sučelju, dogodit će se događaj *onEditCandidateAdded* koji će prouzročiti da se stanje promijeni tako da se broj kandidata poveća za jedan. Kako se stanje mijenja, korisničko sučelje automatski prepoznaje promjenu i dodaje novi vizualni element za novog dodanog kandidata.

```
class CreateBallotStateController extends StateNotifier<CreateBallotStates> {
  CreateBallotStateController(this._web3Service)
      : super(CreateBallotStates.initial());

  final IWeb3Service _web3Service;

  Future mapEventsToStates(CreateBallotEvents events) async {
    return events.map(
        onEditTopicChanged: (value) async {
          state = state.copyWith(
            topic: Topic(topic: value.topic),
          );
        },
        onEditCandidateAdded: (value) async {
          List<Candidate> candidates = state.candidates;
          candidates.add(Candidate(name: ""));
          state = state.copyWith(candidates: candidates);
        },
        onEditDuration: (value) async {
            state = state.copyWith(duration: value.duration!);
        },
        onRemoveCandidate: (value) async {
          state.candidates.removeAt(value.candidateId!.toInt());
          state = state.copyWith(candidates: state.candidates);
        },
        onEditCandidateName: (value) async {
          state.candidates[value.candidateId!.toInt()] =
              Candidate(name: value.candidateName!);
          state = state.copyWith();
          print(state.candidates.toString());
        },
        createBallotBox: (value) async {
          await _createBallotBox(_web3Service.createElection);
        },
```

Sl. 4.19. *Prikaz klase upravljača slučajevima za kreiranje nove glasačke kutije*

4.5. Sloj korisničkog sučelja

Flutter se zasniva na deklarativnom pristupu korisničkog sučelja. To znači da Flutter gradi korisničko sučelje kao odraz trenutnog stanja aplikacije, za razliku od Android i iOS-razvoja gdje bi po potrebi mijenjali vrijednosti komponente korisničkog sučelja, Flutter je dovoljno brz da prilikom promjene stanja ponovno kreira komponente korisničkog sučelja od nule.

Widget je način deklariranja i konstruiranja korisničkog sučelja. Zaslon se kreira slaganjem i gniježđenjem *widgeta* tvoreći *widget* stablo. *Widget* sadrži konfiguraciju korisničkog sučelja za iscrtavanje na zaslonu uređaja. Jednom definirana, konfiguracija je nepromjenjiva za vrijeme izvođenja. Konfiguracija korisničkog sučelja sadrži položaj, stilove i ukrase. Tijekom izvođenja, *widget* u stablu *widgeta* se „napuhuje“ u *Element* koji stvara instancu danog widgeta, a zatim poziva metodu *mount()* za dodavanje elementa u stablo elemenata na određenom mjestu i po potrebi poziva metodu *attachRenderObject()* da bi pridružio sve povezane objekte renderiranja u *RenderView* unutar *RenderTree-a*. Nakon što je element montiran i renderiran, element će se pojaviti na ekranu uređaja i generirati dati *widget*.

Provider je biblioteka pomoću koje se korisničkom sučelju pružaju sve potrebne ovisnosti i uz pomoću koje korisničko sučelje promatra promjene stanja. Promatrajući stanje, *widgeti* se po potrebi, ili točnije, po promjeni stanja mogu mijenjati.

Primjer *providera* koji se koristi na zaslonu za glasovanje je prikazan na slici 4.20. Na osnovu identifikacije glasačke kutije, *provider* vraća instancu klase *VoteStateController* koja u sebi ima definirane događaje i stanja definirana za upravo taj zaslon.

```
25  final voteViewProvider =
26  StateNotifierProvider.family.autoDispose<VoteStateController, VoteStates, Id>((ref, id) {
27    final privateKey = ref
28        .watch(firestoreProvider)
29        .userData
30        .privateKey
31        .valueObject!
32        .fold((l) => throw UnExpectedValueError(l), (r) => r);
33
34    final ethAddress = ref
35        .watch(firestoreProvider)
36        .userData
37        .address
38        .valueObject!
39        .fold((l) => throw UnExpectedValueError(l), (r) => r);
40    return VoteStateController(Web3Service(privateKey, ethAddress), id);
41  });
```

Sl. 4.20. *Prikaz inicijalizacije providera na zaslonu za glasovanje*

Na slici 4.21 se može vidjeti *widget* stablo koje tvori gumb glasanja na zaslonu za glasovanje. Važno je istaknuti događaj *VoteEvents.vote()* koji se poziva prilikom *onPressed* stanja *ElevatedButton*-a. Što znači da će se prilikom pritiska na gumb *vote( )* pokrenuti događaj koji uzrokuje izvršavanje funkcije glasovanja, a naposljetku i promjenu stanja..

```
Container(
  margin: EdgeInsets.only(top: deviceSize.height * 0.005),
  height: deviceSize.height * 0.05,
  width: deviceSize.width * 0.8,
  child: ElevatedButton(
    onPressed: () {
      formEvents.mapEventsToStates(
        const VoteEvents.vote(),
      );
    },
    style: ButtonStyle(
      backgroundColor:
          MaterialStateProperty.all<Color>(primaryColorDark),
      shape: MaterialStateProperty.all<RoundedRectangleBorder>(
        RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10),
        ),  // RoundedRectangleBorder
      ),
    ),  // ButtonStyle
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Container(
          height: deviceSize.height * 0.03,
          margin: EdgeInsets.only(
              left: deviceSize.width * 0.04,
              right: deviceSize.width * 0.02),  // EdgeInsets.only
          child: FittedBox(
            fit: BoxFit.fitHeight,
            child: Text(
              "Vote",
              style: const TextStyle(
                  color: primaryColor,
                  fontSize: 22,
                  fontWeight: FontWeight.bold),  // TextStyle
            ),  // Text
          ),  // FittedBox
        ),  // Container
      ],
    ),  // Row
  ),  // ElevatedButton
),  // Container
```

Sl. 4.21. *Prikaz gumba za glasovanje na zaslonu za glasovanje*

Slika 6.22 prikazuje jednu glasačku opciju unutar liste glasačkih opcija na zaslonu za glasovanje. Unutar *widgeta TextFromField* postoji argument *initialValue* koja je postavljena na vrijednost izbornog kandidata iz stanja. To znači da će se prilikom promjene vrijednosti stanja, *widget*

TextFromField iznova izgraditi kako bi promijenio vrijednost izbornog kandidata na korisničkom sučelju.

```
child: Container(
  margin: EdgeInsets.all(2),
  padding: EdgeInsets.only(
      top: deviceSize.height * 0.01,
      left: deviceSize.width * 0.01,
      bottom: deviceSize.height * 0.01),  // EdgeInsets.only
  decoration: BoxDecoration(
    color: backgroundColorLight,
    borderRadius:
        BorderRadius.all(Radius.circular(30)),
  ),  // BoxDecoration
  alignment: Alignment.topCenter,
  child: TextFormField(
    enabled: false,
    initialValue: formStates.candidates[index],
    textAlign: TextAlign.center,
    textAlignVertical: TextAlignVertical.center,
    maxLines: null,
    decoration: new InputDecoration(
      border: InputBorder.none,
      focusedBorder: InputBorder.none,
      enabledBorder: InputBorder.none,
      errorBorder: InputBorder.none,
      disabledBorder: InputBorder.none,
      prefix: Container(
        child: LinearGradientMask(
          child: Text(
              "${index + 1}.",
              style: TextStyle(
                  fontWeight: FontWeight.bold,
                  color: primaryColor,
                  fontSize: 18),  // TextStyle
            ),  // Text
          ),  // LinearGradientMask
        ),  // Container
        contentPadding: EdgeInsets.only(
            left: 15,
            bottom: 15,
            top: 15,
            right: 15),  // EdgeInsets.only
      ),  // InputDecoration
    ),  // TextFormField
  ),  // Container
```

Sl. 4.22. *Prikaz glasačke opcije na zaslonu za glasovanje*

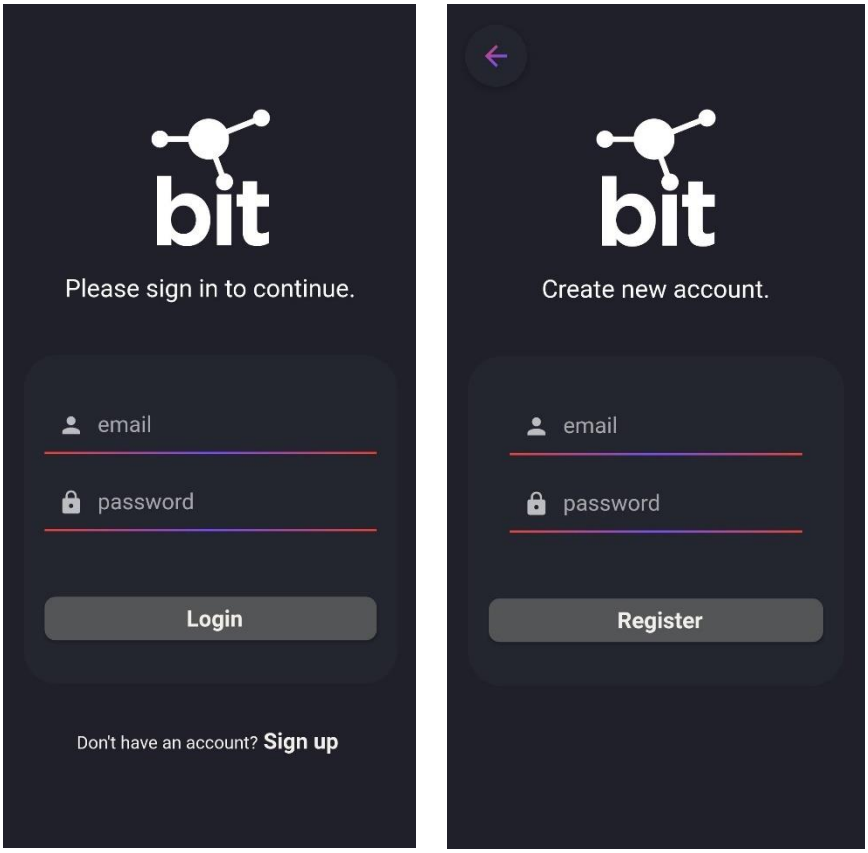
5. KORIŠTENJE APLIKACIJE ZA GLASOVANJE

Ovo poglavlje je namijenjeno za prezentiranje i opis načina korištenja mobilne aplikacije, kako bi korisnik aplikacije za glasovanje imao sve potrebne informacije za uspješnu interakciju s aplikacijom za glasovanje pomoću blockchain tehnologije.

5.1. Autentikacija korisnika

Prvi zasloni koje će korisnik susresti pri prvom pokretanju aplikacije su zasloni autentikacije, prijave i registracije. Korisnik se prijavljuje korisničkim emailom i lozinkom. Email mora biti ispravnog formata dok lozinka mora biti najmanje 7 znamenki i mora sadržavati najmanje jedno veliko slovo, jedno malo slovo i jedan posebni znak. Na dnu zaslona za prijavu, sa slike 5.1, se nalazi tekst namijenjen za korisnike koji još nemaju korisnički račun. Pritiskom na tekst, aplikacije će korisnika dovesti na zaslon za registraciju, gdje će moći kreirati novi račun.

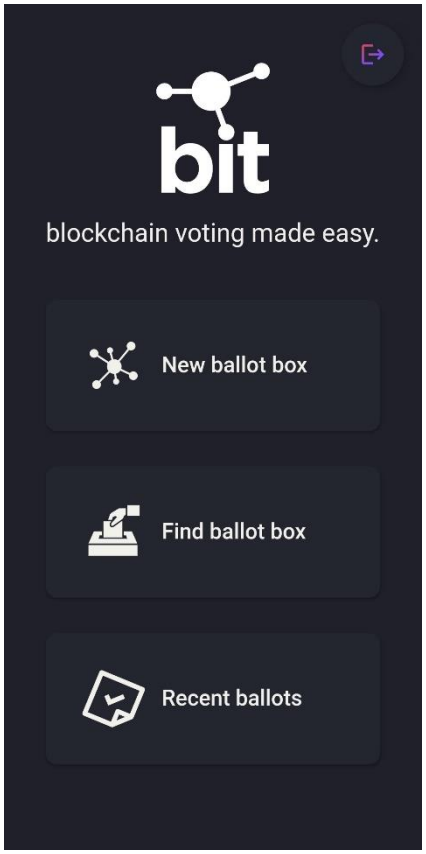
Nakon uspješne prve prijave, korisnik se više neće morati prijavljivati u aplikaciju pri svakom pokretanju. Ako je korisnik ostao prijavljen od zadnje sesije, preskočit će autentikaciju pri pokretanju aplikacije. To će se događati sve dok se korisnik eksplicitno ne odjavi iz aplikacije pritiskom na gumb odjave na glavnom zaslonu koji će biti opisan u nastavku.



Sl. 5.1. *Prikaz ekrana aplikacije za registraciju i prijavu*

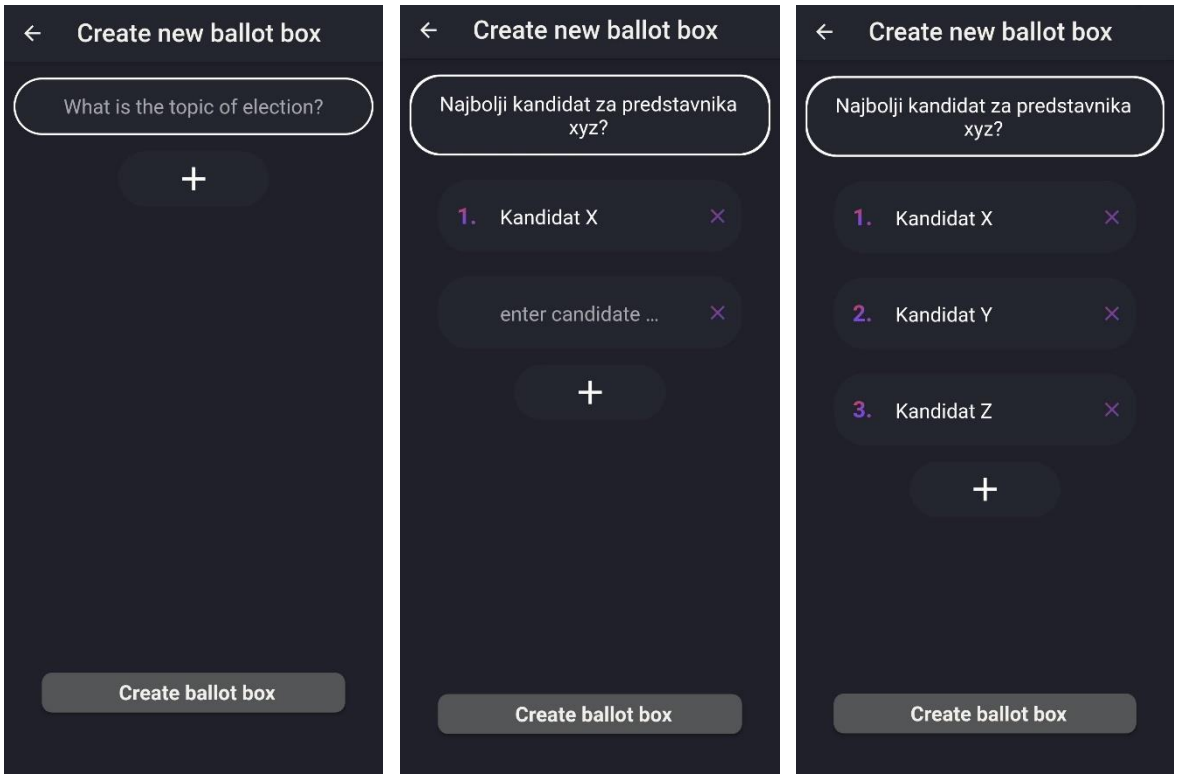
5.2. Kreiranje nove glasačke kutije

Prilikom uspješne autentikacije korisnik će biti navigiran na glavni izbornik (Sl 5.2). S glavnog izbornika korisnik ima opciju kreirati novo glasovanje, glasovati na već postojeće glasovanje i pregledati već izglasana glasovanja. Unutar aplikacije jedno glasovanje(izbori, tema glasovanja) je simbolički nazvan „glasačka kutija“ (engl. *ballot box*), dok „glasački listić“ (engl. *ballot*) predstavlja jedan glas. Uz već spomenute 3 opcije iz glavnog izbornika, korisnik se može i odjaviti iz aplikacije pritiskom na dugme u gornjem desnom kutu. Pritiskom na gumb odjave, korisnik se vraća na zaslon za prijavu s korisničkim imenom i lozinkom.

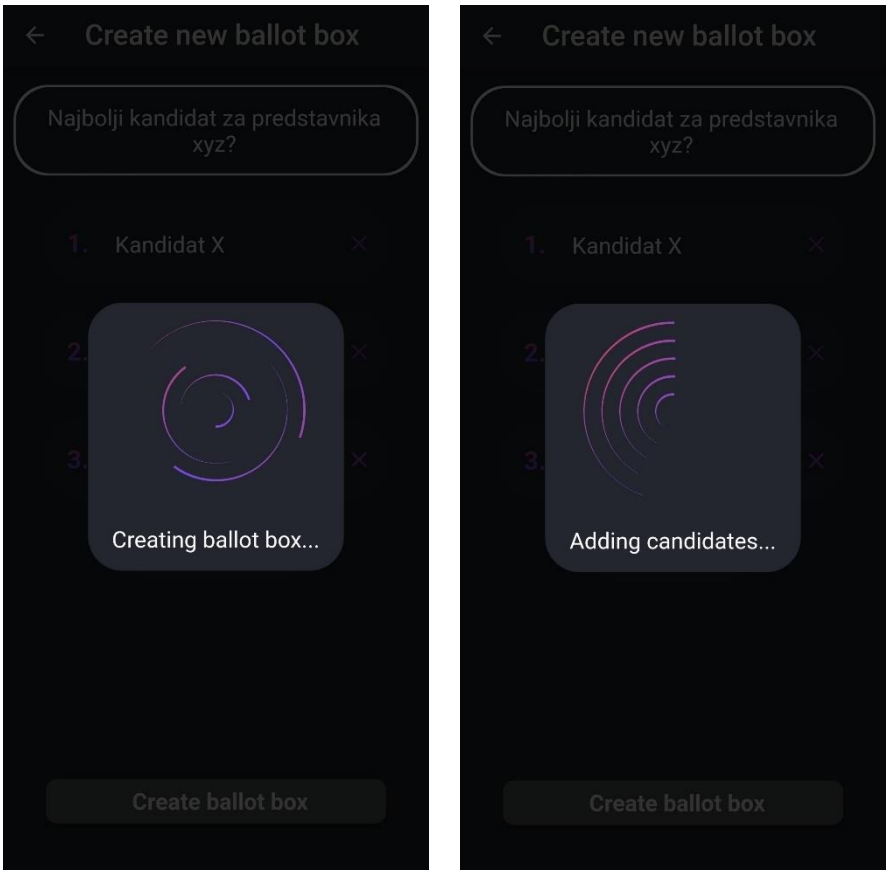


Sl. 5.2. *Prikaz zaslona glavnog izbornika*

Odabirom prve opcije, aplikacije prelazi na zaslon kreiranja novog glasanja. Korisnik sa zaslona za kreiranja nove glasačke kutije može dodati temu glasanja i proizvoljan broj kandidata na izbor. Proces kreiranja nove glasačke kutije započinje pritiskom na *Create ballot box* gumb koji je lociran na dnu zaslona. Proces kreiranja glasačke kutije je podijeljen u više dijelova koji predstavljaju različite transakcije koje se odvijaju na blockchainu. Primjena tih koraka, odnosno transakcija, je vidljiva prilikom čekanja na kreiranje glasačke kutije (Sl 5.4). Prilikom uspješnog kreiranja glasačke kutije aplikacija za glasovanje prelazi na zaslon za dijeljenje glasačke kutije.



Sl. 5.3. *Prikaz zaslona stvaranja nove glasačke kutije*



Sl. 5.4. *Prikaz zaslona pri čekanju na završetak procesa kreiranja glasačke kutije*

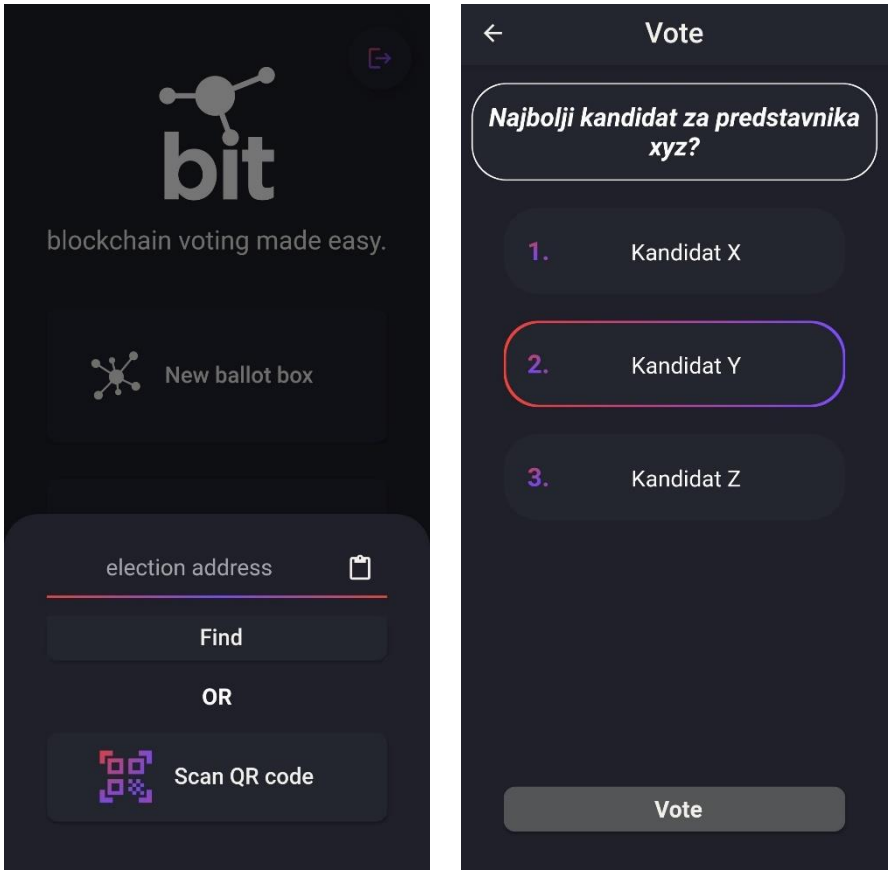
Korisnik generira adresu po završetku kreiranja glasačke kutije. Adresu glasačke kutije je moguće podijeliti ostalim glasačima jednostavnim kopiranjem koda pritiskom na ikonicu za kopiranje ili dijeljenjem QR koda. Isto tako, na zaslonu dijeljenja glasačke kutije (Sl. 5.5), imamo opciju za glasovanje. Pritiskom na gumb *Vote* tvorac glasačke kutije se navigira na zaslon glasovanja, gdje će imati priliku glasovati za svog kandidata.

Sl. 5.5. *Prikaz zaslona za dijeljenje glasačke kutije*

5.3. Glasovanje

Korisnik može glasovati na tri načina. Prvi način je odmah nakon kreiranja glasačke kutije. Korisnik koji je kreirao glasovanje ima ujedno i pravo na glasovanje, tako da neposredno nakon kreiranja glasačke kutije korisnik može i dati svoj glas. Druga dva načina glasanja su za korisnike koji nisu tvorci glasačke kutije. Ostali korisnici glasanju mogu pristupiti unosom adrese glasačke kutije ili skeniranjem QR koda glasačke kutije. Oba će načina korisnika odvesti na zaslon glasovanja. Na zaslonu glasanja je prikazano izborno pitanje s ponuđenim opcijama. Pritiskom na opcije korisnik označava kandidata kojemu želi dati svoj glas. Označavanje kandidata nije konačno, glas se šalje tek kada korisnik pritisne na gumb *Vote* s uvjetom da je jedan od kandidata

označen. Nakon određenog vremena od pritiska na gumb *Vote* i uspješne transakcije, korisnika će aplikacija navigirati nazad na glavni izbornik.



Sl. 5.6. *Prikaz zaslona pretraživanja glasačke kutije*

5.1. Pregled prijašnjih glasačkih listića

Pritiskom na treću opciju pod nazivom *Recent ballots* na glavnom izborniku, aplikacija prelazi na zaslon prijašnjih glasačkih listića. Na spomenutom zaslonu korisnik ima uvid u sva svoja prijašnja glasovanja. Prijašnja glasovanja su prikazana u listi s karticama gdje svaka kartica sadrži izborno pitanje s dodatnim ikonicama koje predstavljaju status izbora. Prilikom pritiska na jednu od kartica, korisnika se navigira na zaslon gdje se mogu vidjeti rezultati glasovanja po kandidatu (Sl.5.7).

Sl. 5.7. *Prikaz zaslona prijašnjih glasačkih listića*

6. ZAKLJUČAK

U ovome trenutku mobilno elektroničko glasovanje, niti uz pomoć blockchain tehnologije nije dovoljno sigurno da bi trajno zamijenilo tradicionalne načine glasovanja. Blockchain tehnologija, iako donosi nove ideje o identitetu, transparentnosti i validaciji, poboljšava segmente koji nisu niti bili najugroženiji. Najveće ranjivosti ovakvog sustava i dalje ostaju elektronički uređaji koji se lako ugroze i nesiguran medij prijenosa glasova u obliku interneta. Takav sustav glasovanja nije spreman postati primaran način glasovanja za masovnu upotrebu po nacionalnim izborima u svijetu, no mogao bi pronaći svoje mjesto kao sekundarna opcija za osobe koje primjerice nisu u prilici pristupati izborima iz zdravstvenih razloga. Takav način glasovanja kao sekundarna opcija za onemogućene državljane bi bio koristan i u Republici Hrvatskoj, no trenutno za takvo što nema nikakvih planova niti zakonske regulative. Internet glasovanje treba pronaći svoje mjesto u situacijama gdje subjektivno nepovoljan rezultat izbora ne utječe na vrlo velik broj ljudi kao što je to slučaj s nacionalnim izborima. Idejno rješenje ove aplikacije se zasniva upravo na tome, kreiranje aplikacije za glasovanje koje će koristiti grupe, vijeća i organizacije u potrebi za jednostavnim rješenjem za glasovanje. Aplikacija pruža funkcionalnost kreiranja novih izbora i glasovanje na aktualnim izborima kojim se pristupa skeniranjem QR koda ili unosom adrese izbora. Aplikacija je otvorena i za proširenja, najviše u vidu integracije vanjskih autentikacija, gdje bi se korisnik u aplikaciju prijavljivao sa svojim studentskim, službenim ili građanskim korisničkim računom.

LITERATURA

[1] L. D. Merkle i M. H. Dunn., Overview of Software Security Issues in Direct-Recording Electronic Voting Machines, [Mrežno]. Dostupno na: https://www.researchgate.net/publication/326981756_Overview_of_Software_Security_Issues_in_Direct-Recording_Electronic_Voting_Machines. [14.9.2021.].

[2] Electronic voting by country, Wikipedia, [Mrežno]. Dostupno na: https://en.wikipedia.org/wiki/Electronic_voting_by_country. [6.9.2021.].

[3] S. Park, M. Specter, N. Narula, L. R. Rivest., Going from Bad to Worse: From Internet Voting to Blockchain Voting, [Mrežno]. Dostupno na: http://people.csail.mit.edu/rivest/pubs/PSNR20.pdf. [6.9.2021.].

[4] Denial of service attack, Wikipedia, [Mrežno]. Dostupno na: https://en.wikipedia.org/wiki/Denial-of-service_attack. [6.9.2021].

[5] Voatz, [Mrežno]. Dostupno na: https://voatz.com/. [6.9.2021.].

[6] Voatz Mobile Voting Platform: An Overview: Security, Identity, Auditability, Voatz, Inc., [Mrežno]. Dostupno na: https://voatz.com/wp-content/uploads/2020/07/voatz-security-whitepaper.pdf. [6.9.2021.].

[7] D. Jefferson, D. Buell, K. Skoglund, J. Kiniry, J. Greenbaum, What We Don't Know About the Voatz "Blockchain" Internet Voting System, [Mrežno]. Dostupno na: https://cse.sc.edu/~buell/blockchain-papers/documents/WhatWeDontKnowAbouttheVoatz_Blockchain_.pdf. [6.9.2021].

[8] M. A. Specter, ,J. Koppel, D. Weitzner, The Ballot is Busted Before the Blockchain: A Security Analysis of Voatz, the First Internet Voting Application Used in U.S., [Mrežno]. Dostupno na: https://internetpolicy.mit.edu/wp-content/uploads/2020/02/SecurityAnalysisOfVoatz_Public.pdf. [6.9.2021.].

[9] Voatz, Apkpure.com, [Mrežno]. Dostupno na: https://iphone.apkpure.com/voatz/com.nimsim.voatz. [12.9.2021.].

[10] Ethereum Developer Documentation, [Mrežno]. Dostupno na: https://ethereum.org/en/developers/docs. [7.9.2021].

[11] Solidity, [Mrežno]. Dostupno na: https://docs.soliditylang.org/ . [7.9.2021]

[12] Truffle Suite, [Mrežno]. Dostupno na: https://www.trufflesuite.com. [7.9.2021].

[13] Firebase, [Mrežno]. Dostupno na: https://firebase.google.com/docs. [6.9.2021].

[14] Flutter Docs, [Mrežno]. Dostupno na: https://flutter.dev/docs. [6.9.2021].

[15] Domain-Driven Design: What is it and how do you use it?, AirBrake [Mrežno]. Dostupno na: https://airbrake.io/blog/software-design/domain-driven-design. [9.9.2021.].

[16] web3dart, pub.dev, [Mrežno]. Dostupno na: https://pub.dev/packages/web3dart. [9.9.2021.].

SAŽETAK

U ovom diplomskom radu je izrađena mobilna aplikacije za glasovanje koristeći blockchain tehnologiju. Aplikacija pruža funkcionalnost kreiranja vlastitih izbora i glasovanje na drugim izborima preko skeniranja QR koda ili unosom adrese glasovanja. Za autentikaciju je korišten Firebase servis dok je za pohranu podataka korišten ethereum blockchain uz pomoć alata Truffle, Ganache, web3 i Solidity programskog jezika. Mobilna aplikacija je kreirana unutar Flutter programskog okvira po DDD arhitekturalnom dizajnu programske podrške. Kroz rad je opisano i trenutno stanje elektroničkog glasovanja i njegove tehnologije, doprinos blockchaina konceptu mobilnog glasovanja i sigurnosne mane ovakvog načina glasovanja.

Ključne riječi: blockchain glasovanje, elektroničko glasovanje, ethereum, mobilna aplikacija, pametni ugovor

ABSTRACT

Title: Voting application using blockchain technology

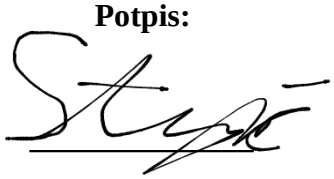
In this master's thesis, a mobile application has been realized using blockchain technology. The application provides the functionality of creating your own elections and voting in other elections by scanning the QR code or entering the election address. Firebase service is used for authentication, while the ethereum blockchain is used for data storage using the tools Truffle, Ganache, web3 and Solidity programming language. The mobile application was created within Flutter software framework by the principles of the DDD architectural design. The paper also describes the current state of electronic voting and its technologies, the contribution of blockchain to the concept of mobile voting and the security flaws of this way of voting.

Key words: blockchain voting, electronic voting, ethereum, mobile application, smart contract

ŽIVOTOPIS

Tomislav Stipanić je rođen 15. prosinca 1996. u Osijeku. Osnovnu školu završava u Bilju 2011. godine i upisuje Elektrotehničku i prometnu školu u Osijeku, smjer Elektrotehničar.
2015. godine Upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. U akademskoj godini 2019./2020. upisuje diplomski studij, smjer Programsko inženjerstvo, na istom fakultetu.

Potpis:



PRILOZI

PRILOG 1. Dokument diplomskog rada

PRILOG 2. Diplomski rad u PDF formatu

PRILOG 3. Programski kod aplikacije za glasovanje koristeći blockchain tehnologiju