Razvoj pametnih ugovora i poslužiteljskog sloja Web3 aplikacije za upravljanje ulaznicama

Blekić, Sandi

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

SANDI BLEKIĆ

RAZVOJ PAMETNIH UGOVORA I POSLUŽITELJSKOG SLOJA WEB3 APLIKACIJE ZA UPRAVLJANJE ULAZNICAMA

Završni rad

Pula, rujan, 2023.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

SANDI BLEKIĆ

RAZVOJ PAMETNIH UGOVORA I POSLUŽITELJSKOG SLOJA WEB3 APLIKACIJE ZA UPRAVLJANJE ULAZNICAMA

Završni rad

JMBAG: 0303075866, redoviti student
Studijski smjer: Informatika

Predmet: Web Aplikacije
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija
Mentor: doc. dr. sc. Nikola Tanković

Pula, rujan, 2023.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Sandi Blekić, kandidat za prvostupnika Informatike, ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljeni način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

U Puli, rujan, 2023. godine

IZJAVA O KORIŠTENJU AUTORSKOGA DJELA

Ja, Sandi Blekić, dajem odobrenje Sveučilištu Jurja Dobrile u Puli, nositelju prava korištenja, da moj završni rad pod nazivom „*Razvoj pametnih ugovora i poslužiteljskog sloja Web3 aplikacije za upravljanje ulaznicama*“ upotrijebi da tako navedeno autorsko djelo objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te preslika u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu sa Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama. Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

Student

U Puli, rujan, 2023. godine

Sadržaj

1 Uvod

Web3 je pojam koji je usko povezan s evolucijom blockchain tehnologije koju je pridonio Ethereum protokol. Uvođenjem novog oblika blockchaina koju pokreće Ethereumovo virtualno računalo (eng. *Ethereum virtual machine*) i koncepta pametnih ugovora, Ethereum je otvorio vrata za razvoj nove vrste web aplikacija takozvanih „dApps“ iliti decentralizirane aplikacije.

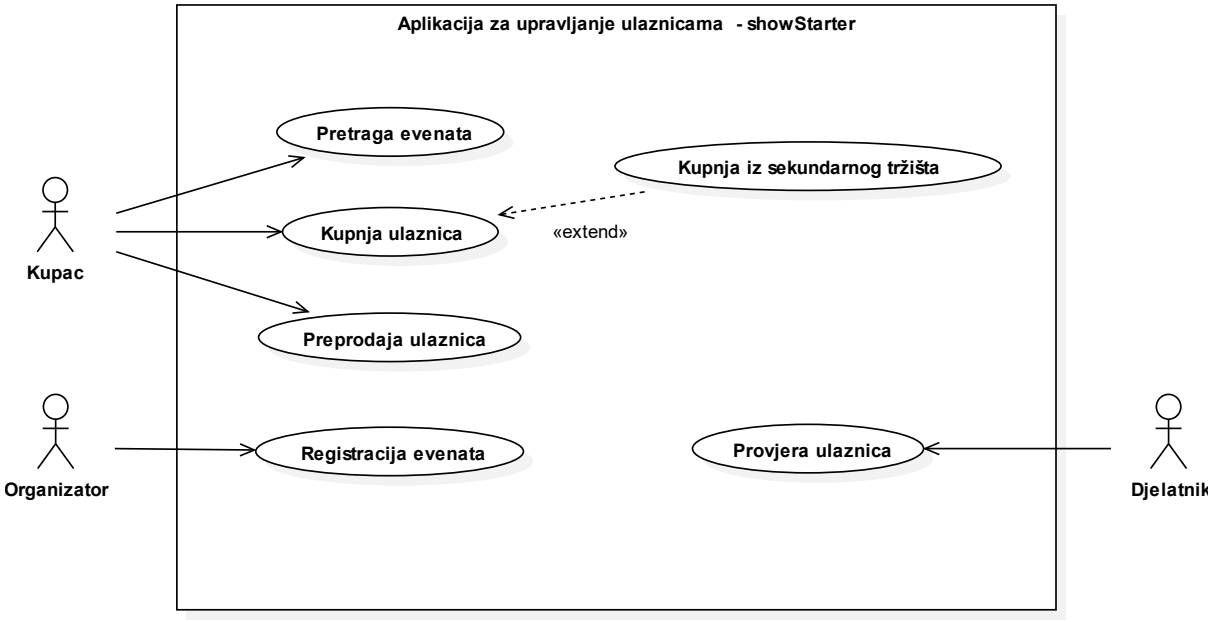
Namjera ovog rada je istraživati moguću implementaciju Web3 tehnologije, specifično koncept ERC-721 standarda za nedjeljive žetone (eng. *non-fungible token*, skraćeno NFT), u kontekstu prodaje ulaznica. Za navedenu svrhu, razvijena je web3 aplikacija "showStarter", koja koristi Polygon-Ethereum platformu radi provedbu prodaje ulaznica u ime organizatora koncerata. Web3 dio aplikacije se sastoji od pametnih ugovora po imenu "EventFactory" i "EventImplementation" koji se koriste za dodavanje novih događaja i za upravljanje logikom prodaje ulaznica. Primarna funkcija pametnih ugovora je organizatoru omogućiti upravljanje sekundarnog tržišta, te kupcu pružiti sigurni kanal kupnje i prodaje ulaznica na istim.

Rad se sastoji od dva dijela. U prvom djelu promatra se arhitektura aplikacije i opisuju se korištene tehnologije, dok u drugom djelu se razrađuje implementacija.

Implementacija projekta je podijeljena u dvije cjeline, odnosno frontend i backend, te se može pregledati na sljedećim Github repozitorijima:

- Frontend: https://github.com/sblekic/zavrsni-frontend
- Backend: https://github.com/sblekic/zavrsni-backend

2 Definiranje zahtjeva



Slika 1. UML Dijagram korištenja

Cilj ove aplikacije je posjetiteljima koncerata pružati sigurnu platformu za kupnju i preprodaju ulaznica a organizatorima omogućiti kontrolu nad sekundarnim tržištem. Za tu svrhu nudi se mogućnost kreiranja događaja, otkrivanja događaja, kupnje i preprodaje ulaznica i provjera ulaznica. Za korištenje navedenih usluga potreban je uređaj koji može pristupiti internetu, podržava pregled web stranica i instalaciju softvera „Metamask“. Sustav će biti korišten od strane organizatora događaja, kupaca i djelatnika.

Organizator događaja može kreirati novi koncert postavljanjem podataka o istom: naziv, prostor održavanja, ime izvođača, datum i vrijeme početka/kraja, kategorija ulaznica, količina dostupnih ulaznica i cijena. Sustav omogućava prodaju karata isključivo za koncerte u trajanju jednog dana, to jest ne uključuje događaje poput festivala. Kreacija novog koncerta ne uključuje proces rezerviranja prostora održavanja i sklapanja ugovora sa izvođačima.

Kupac može pretražiti koncerte ovisno o mjestu održavanja ili imenu izvođača, kupiti ulaznice direktno od organizatora ili na sekundarnom tržištu i prodati kupljene ulaznice. Prodaja ulaznica sa strane korisnika je dopuštena ako je postavljena cijena ista ili manja od nominativne.

Za provjeru valjanosti ulaznice, web stranica nudi djelatnicima funkciju skeniranja QR koda.

Radi pružanja kvalitetnog korisničkog iskustva aplikacija mora nuditi intuitivno korisničko sučelje koje se prilagođava veličini ekrana na kojemu se prikazuje. Naknadno, sustav mora biti dostupan u bilo koje doba sa minimalnim zastojima radi održavanja.

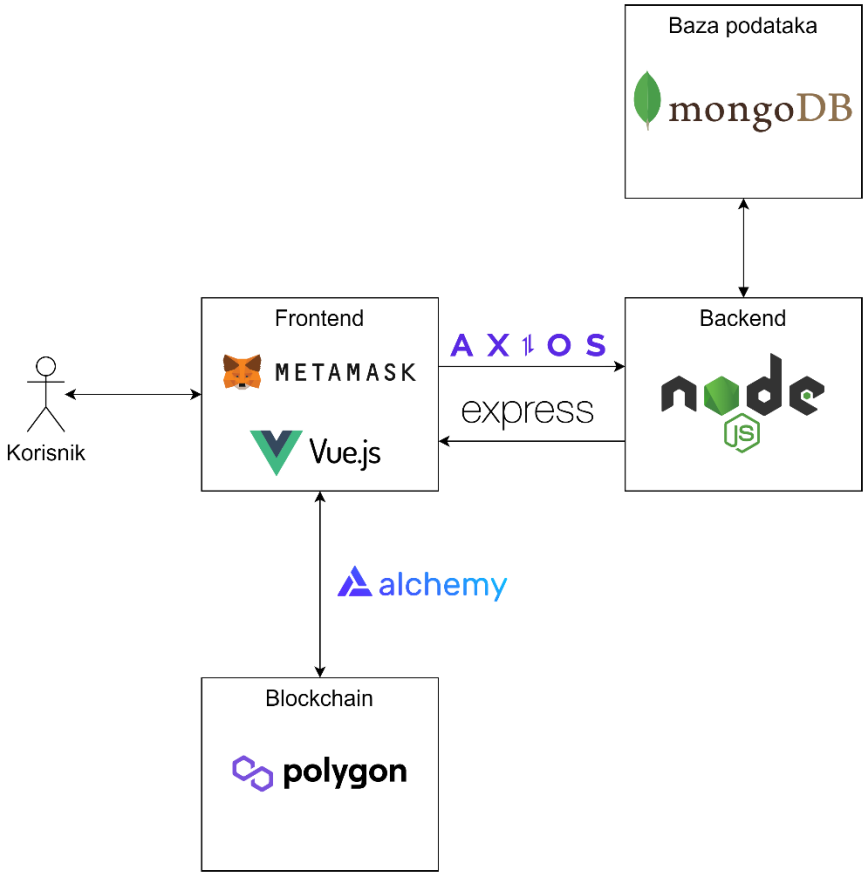
3 Arhitektura

Web aplikacija „showStarter“ razvijena je prema arhitekturi koja je vođena događajima (eng. *Event-driven architecture*). Navedena arhitektura je odabrana zbog toga što se stanje baze podataka mijenja u odnosu na promjene na blockchainu. Aplikacija se dijeli na dva dijela, frontend i backend.

Frontend aplikacije je razvijen u Vue.jsu, koji garantira reaktivnost sučelja, upravlja navigacijom web stranice te pohranjuje njeno globalno stanje pomoću Pinia skladišta. Za pojednostavljeno upravljanje izgledom i prilagodljivosti stranice na ekrane različitih veličina korišten je Bootstrap. Za uspostavu komunikacije sa backendom koristi se HTTP klijent Axios. Veza prema blockchainu se ostvaruje putem platforme „Alchemy“ koja omogućava pristup na testnoj mreži Mumbai-Polygon blockchaina. Biblioteka Ethers se koristi radi intuitivnog Javascript sučelja koji olakšava interakciju sa blockchainom.

Backend je implementiran u Node.jsu prema REST arhitekturnom stilu i koristi se prvenstveno kao veza sa bazom podataka te za posluživanje spremljenih podataka prema klijentu. Završne točke poslužitelja su implementirane pomoću paketa Express, dok se Web3 autentifikacija korisnika provodi putem Moralis biblioteke u kombinaciji sa jsonwebtoken paketom. Poslovna logika sustava je definirana u pametnim ugovorima koji su razvijeni i razmješteni (eng. *deployed*) na Mumbai-Polygon

mreži korištenjem razvojnog okruženja Hardhat. Trajna pohrana stanja sustava se izvršava na hibridnom načinu. Podaci koji su esencijalni za izvršavanje poslovne logike se spremaju na blockchainu, dok se podaci koji moraju biti često dohvaćeni spremaju u bazi podataka MongoDB.



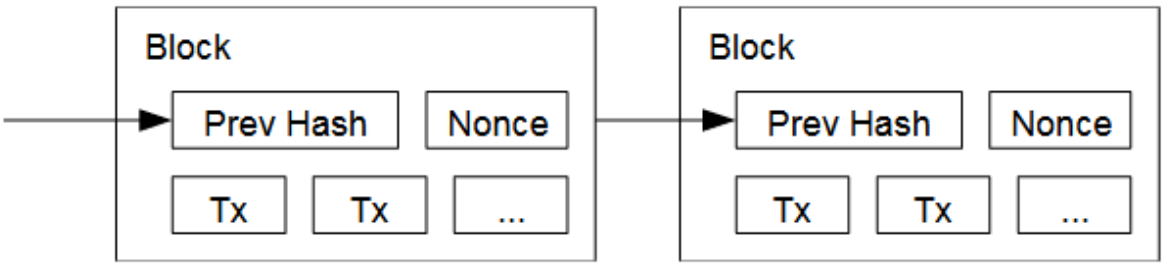
Slika 2. Arhitektura projekta

3.1 Blockchain – tehnologija ulančanih blokova

Blockchain tehnologiju Bashir definira na sljedeći način: „Blockchain je kriptografski osigurana glavna knjiga koja je distribuirana u mreži ravnopravnih dionika (peer-to-peer network), na koju se podaci mogu isključivo dodati, koja je nepromjenjiva i koja se može ažurirati isključivo pomoću implementiranog mehanizma konsenzusa koji sudionici mreže moraju ispoštovati“ [1, str. 16].

Postoji mnogo različitih definicija i tumačenja blockchain tehnologije, no za potrebe ovog rada, tehnologija ulančanih blokova je definirana kao distribuirana baza podataka u kojoj ne postoji središnji entitet autoriteta i u kojoj su podaci nekrivotvorivi i nepromjenjivi.

Za razliku od običnih baza podataka gdje se podaci obično spremaju u tablicama, blockchain, kako i samo ime sugerira, pohranjuje zapise u blokovima koji su poredani u kronološkom redoslijedu i koji su povezani pokazivačima koji sadrže referencu na prijašnjem bloku u obliku kriptografskog *hasha*.



Slika 3. Stuktura ulančanih blokova [2, str.3]

Hash vrijednost bloka je jedinstvena vrijednost u heksadecimalnom obliku koja ga identificira. Hash vrijednost dobiva se putem kriptografske funkcije za sažimanjem (engl. *cryptographic hash function*) koja od spremljenih zapisa i reference na prijašnjem bloku generira niz znamenki fiksne duljine. Bitno svojstvo hash funkcije je njena deterministička priroda, gdje isti ulaz podataka rezultira u istu hash vrijednost. U kontekstu blockchaina bi to značilo da svaki pokušaj promjene podataka mijenja i hash vrijednost bloka. S obzirom na to da svaki blok sadrži referencu na hash vrijednost prijašnjeg bloka, moguće je detektirati pokušaj promjene na efikasan način, pošto se referenca neće više podudarati. Navedena kriptografska tehnika predstavlja zapravo figurativni lanac koji povezuje blokove i čini blockchain otpornim na modifikacijama.

No ovo rješenje samo po sebi nije dovoljano kako bi se osigurala nepromjenjivost podataka. U slučaju da maliciozni akter želi promijeniti već upisani zapis, zbog nastale promjene hash vrijednosti trebao bi također zamijeniti sve blokove koje se nalaze iza promijenjenog bloka.

Kako bi se to spriječilo, treba postojati entitet ili mehanizam koji definira validnost podataka u bloku i odlučuje koji će se blokovi pridodati lancu. S obzirom na to da je

blockchain decentralizirana baza podataka, ne postoji centralni autoritet koji upravlja stanjem iste, te zbog toga treba postojati način na kojem svi članovi mreže mogu postići dogovor oko dodavanja novog bloka. Za tu svrhu, svaka implementacija blockchaina mora sadržati algoritam konsenzusa. Navedeni mehanizam izvršava verifikaciju i validaciju blokova i očuva integritet mreže, odnosno pobrine se o tome da ne postoji specifičan član mreže koji ima kontrolu nad njom [1, str 35].

Prva implementacija blockchain tehnologije je Bitcoin. Razvijena od anonimnog pojedinca ili grupe Satoshi Nakamoto, Bitcoin je prva digitalna valuta koja je izdana i prenesena na decentraliziranoj mreži ravnopravnih dionika. U ovoj implementaciji, blockchain se koristi kao javna decentralizirana baza podataka koja pohranjuje novčane transakcije svojih korisnika, te tako istovremeno prati stanje svih računa [3, str 7]

3.2 Ethereum

Prvi put predstavljen od strane Vitalika Buterina 2013. i lansiran na tržištu 2015. godine [4], Ethereum je softver koji proširuje primjenu blockchain tehnologije, koja je do tada bila uglavnom korištena za implementaciju decentraliziranih platforma za elektroničko plaćanje, i uvede platformu za izgradnju decentraliziranih aplikacija.

Glavna razlika između Bitcoina i Ethereuma je to što Bitcoin prvenstveno koristi blockchain za pohranu novčanih transakcija, dok Ethereum omogućava pohranu i izvedbu računalnog koda.

Na najjednostavniji način Ethereum se može opisati kao blockchain sa ugrađenim računalom [5]. Navedeno računalo se odnosi na Ethereumov virtualno računalo (eng. *Ethereum Virtual Machine - EVM*). EVM je skup umreženih računala koja se nazivaju čvorovi (eng. *nodes*). Da bi se računalo smatralo kao dio Ethereum mreže, odnosno čvorom, potrebno je da pokreće instancu EVM-a putem klijentskog softvera koji implementira Ethereumov protokol [6]. Stanje EVM-a je pohranjen na svakom čvoru te se svaka promjena nad stanjem izvrši u dogovoru sa svim sudionicima. Svaki čvor može EVMu dodati nove funkcionalnosti u obliku pametnih ugovora, koji

će biti dostupni drugim članovima mreže. Zbog navedenih karakteristika, EVM poprima naziv svjetskog računala.

S obzirom na to da svaki sudionik Ethereum mreže može pohraniti svoje programe i pozivati funkcije svojih i drugih programa u mreži, nužno je da postoji mehanizam koji štiti Ethereum od zastoja (beskonačne petlje u kodu, spam transakcija). Za tu svrhu, Ethereum koristi koncept goriva (eng. *gas*). Gorivo je mjerna jedinica koja reprezentira količinu resursa koja je potrebna za izvršavanje pojedinih operacija na Ethereumu. Kao i u stvarnom svijetu, gorivo ima svoju cijenu, koja je definirana u Ethereumovoj kriptovaluti ether. Gorivo se koristi za izvršavanje transakcija te u praksi služi kao naknada za verifikaciju i validaciju transakcije i kao sigurnosni mehanizam koji zaustavlja transakciju u slučaju da nema dovoljno goriva.

3.3 Pametni ugovori

Pametni ugovor (eng. *Smart contract*) je program koji se može izvršiti na Ethereumov blockchainu. Njegovo je stanje i ponašanje pohranjeno na specifičnu adresu blockchaina te se zbog toga ne može naknadno promijeniti [7]. Ono što ga čini pametnim naspram običnog ugovora je to što mu nije potreban posrednik koji će provesti definirane odredbe ako ih stranka ne ispoštuje. Odredbe pametnog ugovora su regulirane njegovim kodom, te kada su uvjeti ispunjeni provedba se realizira na automatizirani način. Termin je uveden 1994. godine od strane Nicka Szaba u radu „Smart Contracts“ [8]. U njegovom najjednostavnijem obliku, pametni ugovor se može usporediti sa samoposlužnim aparatom, u smislu da za određene ulazne parametre garantira deterministične rezultate [9].

Pametni ugovori su ključna komponenta za razvoj decentraliziranih aplikacija (skraćeno „dApp“), te u tom kontekstu se ponašaju poput aplikacijskih programskih sučelja čije su krajnje točke (eng. *endpoints*) javno izložene. Da bi se osigurala kompatibilnost između decentraliziranih aplikacija, kreirani su standardi pod imenom „‘Ethereum Request for Comments“ (skraćeno ERC). Iako Ethereum dopušta kreaciju pametnih ugovora koja ne prate određeni standard, pametna je ideja izbjegavati po-

novni izum kotača svaki put kada se želi kreirati novu kriptovalutu, te definirati standardno sučelje koje garantira interoperabilnost decentraliziranih aplikacija na blockchainu.

Najpopularniji standardi za pametne ugovore su ERC-20 i ERC-721 koji definiraju sučelje za djeljive (eng. *fungible tokens*) i nedjeljive žetone (eng. *non-fungible tokens*, skraćeno NFT).

ERC-20 se odnosi na standard za kreaciju kriptovaluta, koje su po prirodi djeljive. Izraz „djeljiv“ se odnosi na djeljivost žetona u manjim cjelinama i na činjenicu da su žetoni te određene valute međusobno identični odnosno jednakovrijedni.

ERC-721 formalizira koncept vlasništva određene imovine. Za razliku od kriptovaluta, svaki žeton je jedinstven te se razlikuje od drugih svojim karakteristikama, koja mogu biti pohranjena na blockchainu u obliku meta podataka ili poveznice koja vodi do meta podataka. NFTjem može se reprezentirati imovinu iz stvarnog života, digitalnu umjetnost, ulaznicu, decentraliziranu Internet domenu, itd.

3.4 Programski jezik Solidity

Solidity je visoko razinski (eng. *high-level*, ljudski čitljiv), objektno orijentirani programski jezik, namijenjen za implementaciju pametnih ugovora [10].

Na razvoju ovog jezika utjecali su dobro poznati programski jezici poput Pythona i Javascripta, ali u sintaksi Solidityja će se najviše prepoznati kod koji je sličan C++u. Sličnosti sa C++om se najviše primjećuju u sintaksi za deklariranje varijabli, for petlja i preopterećivanje funkcija [11].

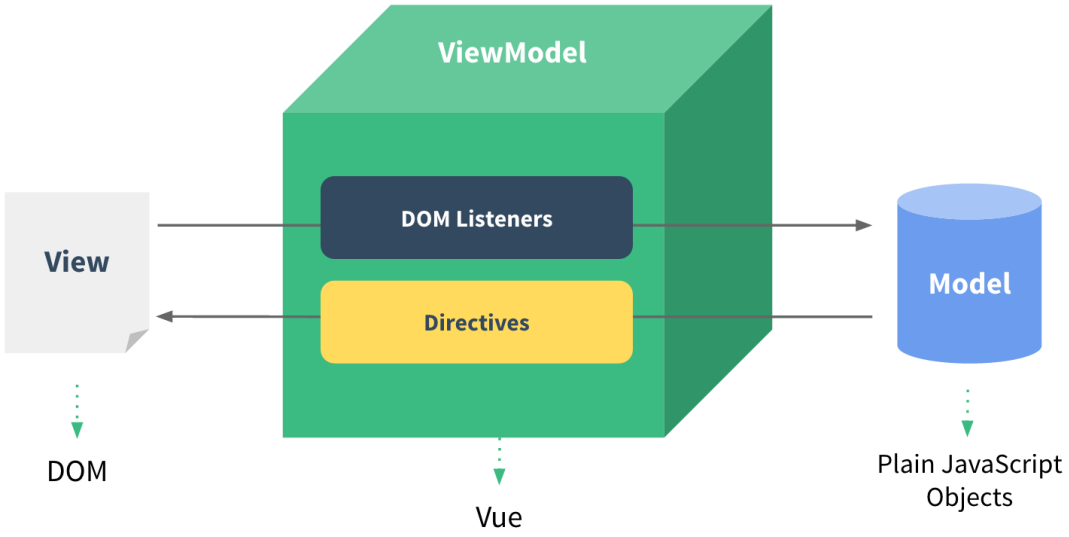
Pametni ugovori napisani u Solidityju su kompatibilni sa svim blockchain platformama koja podržavaju Ethereumovo virtualno računalo. Da bi se programi napisani u Solidityju mogli pohraniti na blockchainu, potrebno ih je prevoditi (eng. *compile*) u bajt-kod koji EVM može interpretirati te izvršiti [12]. Bajt-kod je niz heksadecimalnih znakova koji predstavlja instrukcije, odnosno operacijske kodove (eng. Opcode) koje EVM može izvršiti [6].

Osim bajt-koda, proces kompajliranja generira aplikacijsko binarno sučelje (eng. *ABI – Application Binary Interface*). ABI je sučelje koji na standardizirani način izlaže funkcionalnosti pametnog ugovora.

3.5 Vue.js

Vue je progresivni aplikacijski okvir za izradu interaktivnih korisničkih sučelja, napisan u programskom jeziku JavaScript [13]. Izrađen od Evan You, Vue je nastao kao alat za brzo prototipiranje grafičkih sučelja [14]. Glavni fokus ovog okvira je prezentacijski sloj aplikacije, specifično PogledModel dio Model-Pogled-PogledModel arhitekture (eng. *Model-View-ViewModel*).

U Vue implementaciji MVVM arhitekture, Model je običan Javascript objekt koji sadrži dinamičke podatke koji će biti prikazani korisniku, Pogled reprezentira DOM (eng. Domain Object Model) odnosno ono što je vidljivo korisniku, dok je PogledModel Vue instanca koja ih povezuje i usklađuje [15].



Slika 4. MVVM arhitektura Vuea [15]

Dinamičko prikazivanje podataka, odnosno reaktivnost pogleda, postiže se korištenjem vue direktivama, koje manipuliraju HTML strukturom i slušateljima događaja, koji modificiraju stanje modela ovisno o interakciji korisnika sa stranicom.

Ključna značajka Vuea jest njegova modularna priroda zbog koje mu se pridodaje atribut progresivnog okvira. Vrlo lako se adaptira potrebama određenog projekta, u smislu da ga se može koristiti samo za njegovu primarnu funkciju te po potrebi mu postepeno nadodati funkcionalnosti u obliku dodataka (eng. *plugin*) kako opseg projekta raste [16]. Primjeri spomenutih dodataka su Vue-Router, koji upravlja navigacijom web aplikacije i Pinia skladište (eng. *store*) koji upravlja globalnim stanjem.

Prednost korištenja ovog okvira jest intuitivan način na kojem je strukturiran kod. Dijelovi korisničkog sučelja je moguće podijeliti u posebnim jedinicama zvanim komponente (eng. Single-File Components). Vue komponente su datoteke posebnog „*.vue“ formata koje na jednom mjestu grupiraju strukturu (HTML), dizajn (CSS) i logiku (JavaScript) pojedinog dijela sučelja [17]. Pogodnosti navedenog načina strukturiranja projekta su mogućnost ponovnog korištenja koda, poboljšana čitljivost, olakšano održavanje i testiranje koda, bolja skalabilnost, kraće vrijeme razvoja [18].

3.6 Bootstrap

Bootstrap je skup alata (eng. *toolkit*) otvorenog koda koji omogućava razvoj korisničkih sučelja na brz i jednostavan način [19]. Stvoren je 2011. godine od strane tadašnjih Twitter zaposlenika Mark Otto i Jacob Thornton zbog potrebe standardiziranja frontend alata u cijeloj tvrtki [20]. Zamišljen je za razvijanje responzivnih web stranica, odnosno stranice koje prilagođavaju sadržaj sukladno sa rezolucijom zaslona na kojem su prikazane. Navedena značajka je postignuta korištenjem mrežnog sustava (eng. *grid system*) koji koristi spremnike (eng *containers*), retke i stupce za raspored i poravnanje sadržaja [21]. Bootstrap nudi biblioteku gotovih komponenti sa predefiniranim CSS klasa, koja osiguravaju dosljedan izgled u svim dijelovima stranice.

3.7 Node.js

Node.js je izvršno okruženje (eng. *runtime environment*) otvorenog koda koji omogućava pokretanje Javascript koda izvan preglednika.

Središnja komponenta koja zapravo prevađa i izvršava Javascript kod je Google Chromeov V8 *engine*. Javascript je obično interpretirani jezik, no zbog boljih performansi izvršavanja, V8 koristi JIT (eng. *Just-in-time)* metodu prevađanja koda, koja prevodi kôd tijekom samog izvršavanja istog.

Node.js se uglavnom koristi za pisanje poslužiteljskih programa, te činjenica da izvršava Javascript jezik predstavlja veliku prednost za razvijače web aplikacija, pošto omogućava korištenje istog jezika za razvijanje klijentske i poslužiteljske strane aplikacije.

Glavna značajka Node.jsa je njegova neblokirajuća i asinkrona paradigma programiranja, koja omogućava višestruka i istovremena spajanja na poslužitelja bez kreiranja novih dretva.

Ključni element koji je garantirao uspjeh ovog softvera je *npm* (eng. *Node Package Manager*) upravitelj paketa, koji omogućava preuzimanje gotovih Javascript programa i upravljanje njihovim ovisnostima o drugim programima. U trenutku pisanja ovog rada, registar npm paketa se sastoji od više od dva milijuna paketa [22].

3.8 Express.js

Express je minimalistički programski okvir za Node.js koji se koristi za razvijanje web aplikacija [23]. Glavna mu je svrha pojednostaviti proces kreiranja poslužitelja web stranica. U suštini, Express apstrahira složenost funkcionalnosti Nodeovog HTTP modula i programeru izlaže aplikacijsko programsko sučelje. Zbog toga, programer ne mora iz početka implementirati standardnu logiku koja se bavi rukovanjem HTTP protokola, te može uložiti svoje vrijeme u izgradnju poslovne logike.

Najbitnije komponente koje Express nudi su usmjerivač (eng. *router*) i middleware funkcije. Usmjerivačem se definira ponašanje web aplikacije kada klijent pošalje

HTTP zahtjev na određenu adresu, odnosno rutu, poslužitelja. Ovisno o HTTP operaciji, definiraju se funkcije rukovanja (eng. *handlers*) koje će se izvesti za tu određenu rutu. Middleware je modul koji apstrahira uobičajene funkcionalnosti poput autentifikacije, raščlanjivanje zahtjeva, obrada grešaka, raščlanjivanje kolačića i primjenjuje ih prije izvršavanja funkcije rukovanja na određenoj ruti.

3.9 MongoDB

MongoDB je program koji spada pod kategorijom ne-relacijskih baza podataka (NoSQL). Baza podataka posebno je dizajnirana za upravljanje podacima koji nemaju fiksnu strukturu i koji se možda neće uredno uklopiti u tradicionalne tablične strukture relacijskih baza podataka.

Obilježje koje definira MongoDB je njegova arhitektura orijentirana na dokumente. U MongoDBu su podaci organizirani u zbirke (eng. Collections), od kojih svaka sadrži više dokumenata. U usporedbi sa relacijskim bazama podataka, zbirke se mogu poistovjetiti sa tablicama, a dokumenti sa redcima. Po strukturi, dokumenti podsjećaju na datoteke JSON (eng. Javascript Object Notation) formata, ali su zapravo spremljeni u binarnom formatu po imenu BSON (eng. Binary JSON) koji je optimiziran za pohranu i dohvat podataka.

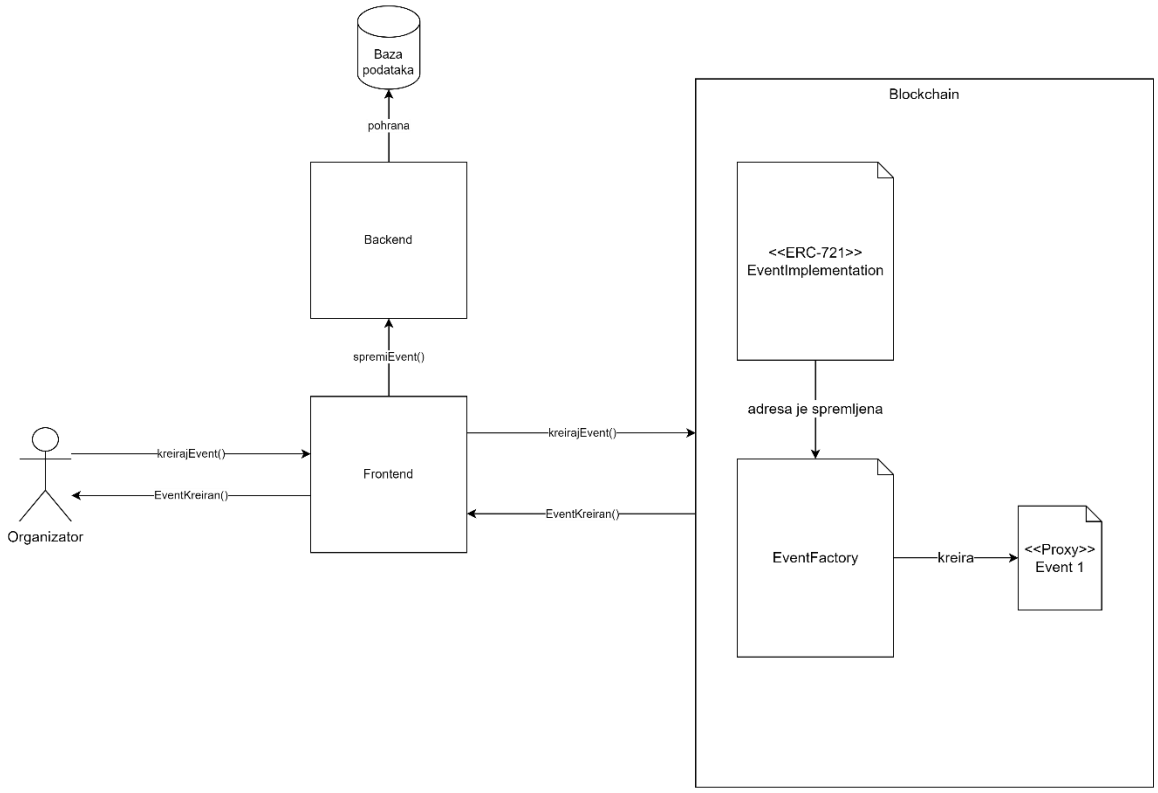
Jedna od značajki MongoDB-a koja se ističe je njegova skalabilnost. Podržava horizontalnu i vertikalnu skalabilnost, dopuštajući dodavanje više poslužitelja ili čvorova kako zahtjevi aplikacije rastu.

4 Implementacija

Srž ovog projekta čini način na kojemu se kreacija događaja i kupovina karata odvija. Za tu svrhu su zaduženi pametni ugovori EventImplementation i EventFactory koji organizatorima događaja omogućavaju upravljanje sekundarnog tržišta, a kupcima pružaju siguran način kupovine i prodaje ulaznica na istim.

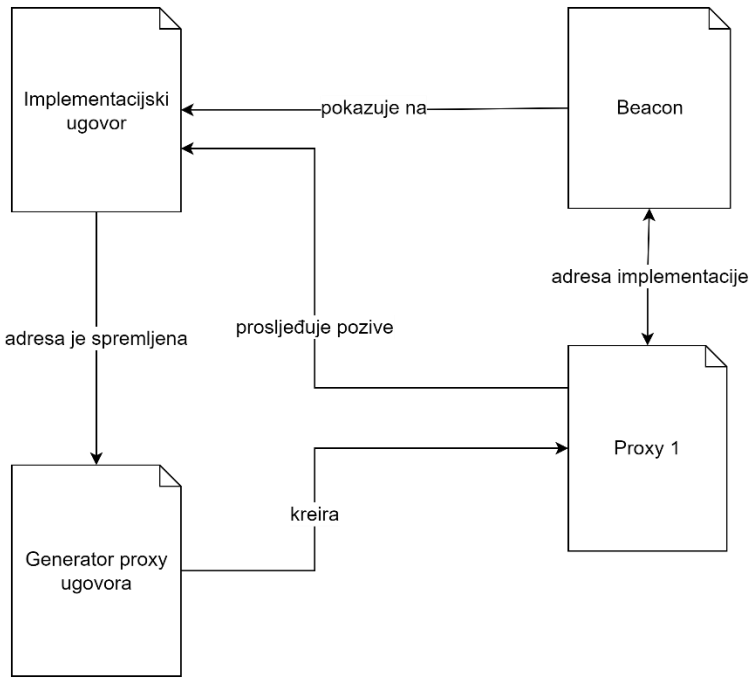
4.1 Registracija evenata

Ovlast za dodavanje evenata imaju organizatori eventa, koji u kontekstu sustava su Ethereum računi koji su reprezentirani nizom heksadecimalnih znakova koji se naziva javna adresa. Za dodavanje novog događaja, na frontendu je organizatoru ponuđen pogled na ruti „/create-event“ koji se sastoji od dva obrasca koji prikupljaju opće informacije o događaju i o ulaznicama koje će se prodati. Kod slanja zahtjeva za stvaranje događaja frontend se prvo spaja sa pametnim ugovorom EventFactory, koji po zahtjevu odobrenih organizatora kreira nove događaje u obliku ERC-721 ugovora. Kada se novi događaj registrira u blockchainu, pametni ugovor pozove Solidity event „*EventCreated*“ koji sadrži adresu pametnog ugovora koji upravlja događajem i obavještava frontend da se transakcija uspješno izvršila. Po primitku navedene obavijesti frontend šalje backendu zahtjev za spremanje podataka o događaju, kako bi budući dohvat tih podataka bio efikasniji. Podaci o događaju pohranjuju se u kolekciji „events“, gdje se između ostalog pohranjuje adresa odgovarajućeg pametnog ugovora, koja se koristi kao jedinstveni identifikator događaja i parametar završne točke „/events/:id“. Kolekcija „events“ se na frontendu koristi za prikaz svih evenata i za filtriranje istih. Apstrahirani proces kreiranja događaja je prikazan na slici 5.



Slika 5. Proces kreiranja događaja

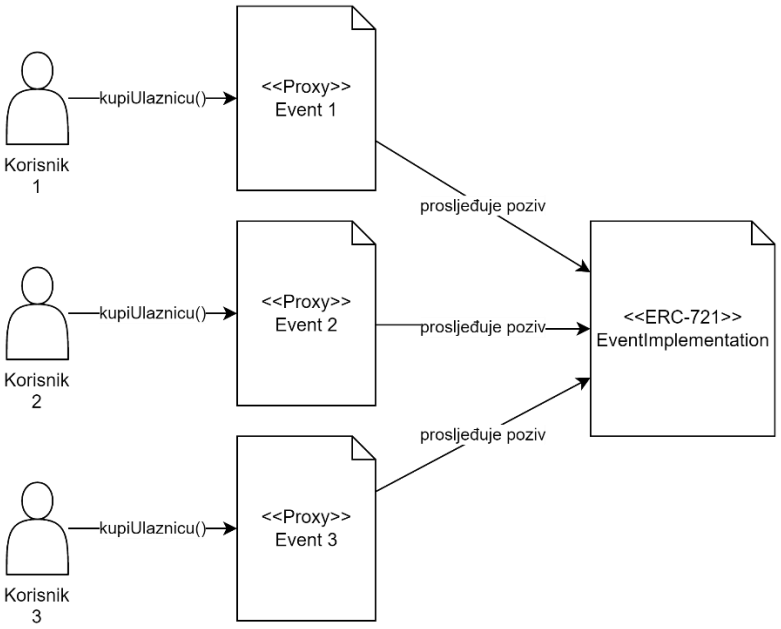
Navedeni proces je implementiran po obrascu posrednika (eng. *proxy pattern*), korištenjem pametnih ugovora „UpgradeableBeacon“ i „BeaconProxy“ iz bilbioteke OpenZeppelin. Glavna prednost ove arhitekture je to što omogućava ažuriranje logike pametnog ugovora bez da se krši nepromjenjivost blockchaina. Navedeno se postiže podjelom pametnih ugovora s obzirom na njihovu ulogu, odnosno na implementacijske i proxy ugovore. Implementacijski ugovor sadrži ponašanje aplikacije a proxy ugovor sadrži njeno stanje. Navedena podjela omogućava kreaciju ugovora koja imaju različita stanja ali isto ponašanje, zbog toga što proxy ugovor delegira izvršavanje logike implementacijskom ugovoru. Ovo je moguće zbog toga što se adresa implementacijskog ugovora pohranjuje u posebnom ugovoru („UpgradeableBeacon“) koji prosljeđuje tu adresu kada proxy ugovor treba delegirati izvršavanje određene funkcije. Ako se želi dodati novu funkcionalnost ili ispraviti grešku u logici, potrebno je razmjestiti poboljšani implementacijski ugovor te spremiti novu adresu u ugovoru „UpgradeableBeacon“.



Slika 5. Grafički prikaz proxy arhitekture

4.2 Kupnja ulaznica

U sklopu projekta svakom događaju je dodijeljen „BeaconProxy“ ugovor koji u kombinaciji sa implementacijskim ugovorom „EventImplementation“ upravlja logikom prodaje ulaznica i pohranjuje podatke o kupoprodaji ulaznica. „EventImplementation“ je ugovor koji prati ERC-721 standard te u tom kontekstu ulaznice su reprezentirane NFTjem, čije je vlasništvo praćeno putem mapping varijable _owners.



Slika 6. Implementacija proxy arhitekture

Kako bi kupovina ulaznica bila dozvoljena kupcu, isti treba prolaziti kroz postupak autentifikacije računa. Postupak se provede na sljedeći način:

1. Backend stvori poruku koja se šalje frontendu na potpisivanje
2. Korisnik potpisuje poruku te se potpis šalje backendu na verifikaciju
3. Od hasha poruke i potpisa generira se javna adresa koja se uspoređuje sa adresom korisnika
4. Ako se adrese podudaraju smatra se da je korisnik stvarni vlasnik računa te je isti prijavljen u aplikaciju

Kod kupovine ulaznica uspostavlja se veza sa pametnim ugovorom koji upravlja događajem te se pozove funkcija „buyTicket“, koja ovisno o kategoriji ulaznice provjerava ako je korisnik uplatio točan iznos i izdaje novi žeton odnosno ulaznicu.

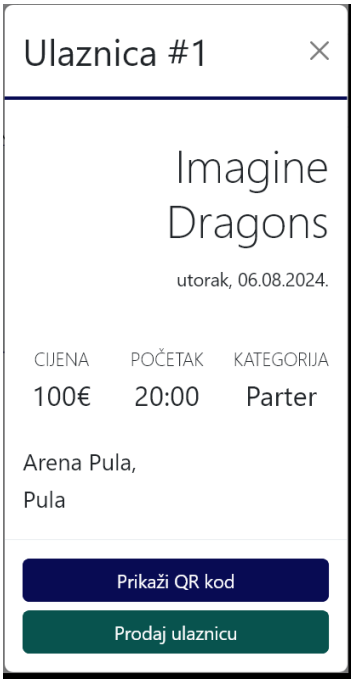
```
    function buyTicket(string memory ticketType) external payable {
        require(tickets[ticketType].supply > 0, "Ulaznice su rasprodane");
        require(
            msg.value == tickets[ticketType].price,
            "Nedovoljan iznos za kupnju ulaznice"
        );
        safeMint(msg.sender);
        tickets[ticketType].supply--;
        emit TicketSale(address(this), _tokenIdCounter.current());
    }
```

Tablica 1. Implementacija funkcije buyTicket() ugovora EventImplementation.sol

Za izdavanje novog žetona koristi se funkcija safeMint, koja u suštini povezuje ulaznicu sa adresom korisnika. Nakon izvršenja transakcije šalje se zahtjev za pohranu podataka o ulaznici na backend aplikacije. Podaci o ulaznici su pohranjeni u kolekciji „tickets“ koja se koristi za filtriranje ulaznica po korisniku, za kreaciju QR koda i za funkcionalnost preprodaje.

4.3 Preprodaja ulaznice

Logikom za preprodaju ulaznica upravljaju funkcije „*sellTicket*“ i „*secondarySale*“ implementacijskog ugovora. Opcija preprodaje je ponuđena u Bootstrapovoj „modal“ komponenti koja reprezentira kupljenu ulaznicu.



Slika 7. Reprezentacija ulaznice na web stranici

Za preprodaju ulaznice, jedini podatak koji se zatraži od korisnika je cijena preprodaje. Za potrebe projekta maksimalna cijena preprodaje je limitirana na originalnu cijenu ulaznice. Misao iza te odluke je da bi se tako moglo obeshrabriti praksu preprodaje ulaznica po znatno višim cijenama (eng. scalping). Nakon uspješne obrade predanog zahtjeva za preprodaju, oglas se prikaže na stranici događaja.

U kontekstu implementacijskog ugovora, preprodaja ulaznice je ekvivalentna prijenosu vlasništva žetona. Time korisnik efektivno daje pametnom ugovoru dopuštenje da u njegovo ime izvrši preprodaju ulaznice. Sa tehničkog aspekta ovaj korak je nužan zbog toga što funkcija „safeTransferFrom“ dopušta prijenos vlasništva žetona isključivo ako zahtjev dolazi od njegovog vlasnika. Zbog te činjenice, u trenutku preprodaje pametni ugovor mora biti vlasnik ulaznice kako bi se prijenos na drugom kupcu mogao izvršiti. Implementacija funkcije za predaju oglasa može se promatrati u tablici 2.

```
    function sellTicket(
        uint256 tokenId,
        uint256 resellPrice,
        string memory ticketType
    ) external {
        require(
            resellPrice <= tickets[ticketType].price,
            "Cijena preprodaje ne smije biti veca od originalne cijene ulaz-
nice"
        );
        idToListedTicket[tokenId] = ListedTicket(
            payable(msg.sender),
            resellPrice,
            true
        );
        safeTransferFrom(msg.sender, address(this), tokenId);
        emit ListedTicketSuccess(tokenId);
    }
```

Tablica 2. Implementacija funkcije sellTicket() ugovora EventImplementation.sol

5 Evaluacija

Jedan od glavnih nedostataka Ethereum mreže je relativno spora brzina obrade transakcija i visoki iznos naknade. S obzirom na to da se u aplikaciji ovog tipa očekuje veliki promet, korištenje Ethereum mreže za izdavanje žetona rezultiralo bi pretjerano visokim naknadama, zbog toga što se cijena goriva usklađuje u odnosu na potražnju za obradu transakcija. Kako bi se to izbjeglo pametni ugovori ove aplikacije su spremljeni na Mumbai mrežu Polygon blockchaina.

Značajka	Ethereum	Polygon
Prosjek transakcija po sekundi	11.2	34.9
Naknada (ERC-20 prijenos)	1.58 €	0.0024 €
Naknada (ERC-721 prijenos)	2.06 €	0.0032 €

Tablica 3. Razlike između Ethereum i Polygon mreže [24]

U tablici 3. prikazane su značajke Ethereum i Polygon mreže koje su relevantne za implementaciju aplikacije. Iz tablice je vidljivo da Polygon mreža može obraditi otprilike duplo više transakcija uz daleko povoljnijim naknadama.

U kontekstu aplikacije promatrana je količina goriva koja je potrebna za kreiranje događaja, kupnju i prodaju ulaznice. Podaci o transakcijama su prikupljeni sa web stranice https://mumbai.polygonscan.com. Polygon Mumbai je mreža koja je namijenjena testiranju pametnih ugovora, te je zbog toga cijena goriva niža od realne. Kako bi se izračunala prosječna cijena transakcije na glavnoj mreži koristit će se cijena jedinice goriva i vrijednost MATIC kriptovalute u eurima u trenutku pisanja ovog rada. Cijena goriva iznosi 97.3 Gwei ($94 * 10^{-9}$ MATIC) a 1 MATIC vrijedi 0.50 eura.

Naknada za kreaciju događaja ovisi o količini znakova koja će biti spremljeni na blockchainu. Na primjer, događaj koji nudi četiri kategorije ulaznica će zahtijevati više goriva od onoga koji nudi samo jednu. U prosjeku je kreacija događaja zahtijevala 506 jedinica goriva, to jest 0.0000246169 eura. Za kupnju i prodaju ulaznica, visina naknade iznosi okvirno 115 i 150 jedinica goriva, odnosno 0.00000559475 i 0.0000072975 eura.

6 Zaključak

Korištenje nedjeljivih žetona kao ulaznice za koncerte nudi nekoliko pogodnosti, kako za organizatore događaja tako i za posjetitelje. Inherentna sigurnost i nepromjenjivost blockchain tehnologije smanjuje rizik krivotvorenih ulaznica i omogućava postavljanje pravila preprodaje ulaznica, s kojima se nastoji savladati problem scalpinga. Osim toga, samo izvršna priroda pametnih ugovora i praćenje vlasništva ulaznica nudi mogućnost automatizirane podjele određenog postotka prihoda od preprodaje ulaznica sa organizatorima događaja i izvođačima.

Iako NFT tehnologija nudi navedena unaprjeđenja, njeno usvajanje u industriji prodaje ulaznica je još uvijek relativno novo, te zbog toga postoje određeni izazovi koje treba razmotriti. Jedan od izazova je činjenica da koncept NFT-a i općenito Web3 tehnologije je još uvijek nejasan široj publici [25] te je potrebno uložiti trud u stvaranju platforme koja je jednako intuitivna kao i njena Web2 varijanta.

7 Literatura

[1] I. Bashir, *Mastering blockchain: distributed ledger technology, decentralization, and smart contracts explained*, Second edition, Fully revised and Updated. in Expert insight. Birmingham Mumbai: Packt, 2018.

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

[3] S. Palladino, *Ethereum for Web Developers: Learn to Build Web Applications on top of the Ethereum Blockchain*. Berkeley, CA: Apress, 2019. doi: 10.1007/978-1-4842-5278-9.

[4] 'A Prehistory of the Ethereum Protocol'. https://vitalik.ca/general/2017/09/14/prehistory.html (Pristupljeno 05-ruj-2023).

[5] 'Intro to Ethereum', *ethereum.org*. https://ethereum.org (Pristupljeno 05-ruj-2023).

[6] Z. Pratap, 'What Are ABI and Bytecode in Solidity?', *Chainlink Blog*, Aug. 17, 2022. https://blog.chain.link/what-are-abi-and-bytecode-in-solidity/ (Pristupljeno 04-ruj-2023).

[7] 'Introduction to smart contracts', *ethereum.org*. https://ethereum.org (Pristupljeno 06-ruj-2023).

[8] 'Smart Contracts'. https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html (Pristupljeno 06-ruj-2023).

[9] 'Nick Szabo -- Smart Contracts: Building Blocks for Digital Markets'. https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html (Pristupljeno 06-ruj-2023).

[10] 'Solidity — Solidity 0.8.21 documentation'. https://docs.soliditylang.org/en/v0.8.21/ (Pristupljeno 04-ruj-2023).

[11] 'Language Influences — Solidity 0.8.21 documentation'. https://docs.soliditylang.org/en/v0.8.21/language-influences.html (Pristupljeno 04-ruj-2023).

[12] 'Learn Solidity: What is the Solidity compiler?' https://www.alchemy.com/overviews/solidity-compiler (Pristupljeno 04-ruj-2023).

[13] 'Vue.js - The Progressive JavaScript Framework | Vue.js'. https://vuejs.org/ (Pristupljeno 03-ruj-2023).

[14] O. Filipova, *Learning Vue.js 2: learn how to build amazing and complex reactive web applications easily with Vue.js*. Birmingham Mumbai: Packt, 2016.

[15] 'Getting Started - vue.js'. https://012.vuejs.org/guide/ (Pristupljeno 04-ruj-2023).

[16] B. Nelson, *Getting to Know Vue.js: Learn to Build Single Page Applications in Vue from Scratch*. Berkeley, CA: Apress, 2018. doi: 10.1007/978-1-4842-3781-6.

[17] 'Single-File Components | Vue.js'. https://vuejs.org/guide/scaling-up/sfc.html (Pristupljeno 04-ruj-2023).

[18] 'A Guide to Component-Based Architecture: Features, Benefits and more', *Maruti Techlabs*. https://marutitech.com/guide-to-component-based-architecture/ (Pristupljeno 04-ruj-2023).

[19] J. S. Jensen, 'The Missing Bootstrap 5 Guide'.

[20] J. Spurlock, *Bootstrap*, F irst edition. Beijing: O'Reilly, 2013.

[21] M. O. contributors Jacob Thornton, and Bootstrap, 'Grid system'. https://getbootstrap.com/docs/5.3/layout/grid/ (Pristupljeno 03-ruj-2023).

[22] 'npm'. https://www.npmjs.com/ (Pristupljeno 03-ruj-2023).

[23] 'Express - Node.js web application framework'. http://expressjs.com/ (Pristupljeno 03-ruj-2023).

[24] "Polygon vs. Ethereum," CoinGecko. https://www.coingecko.com/learn/polygon-vs-ethereum (Pristupljeno: 18-ruj-2023).

[25] 'Web3 and crypto global survey 2023', *Consensys*. https://consensys.io/insight-report/web3-and-crypto-global-survey-2023 (Pristupljeno 08-ruj-2023).

[26] M. Miličević, 'Svijet scalpinga, drevne vještine preprodaje karata', *Muzika.hr*, May 14, 2018. https://www.muzika.hr/preprodaja-ulaznica-bojkot-scalping/ (Pristupljeno 08-ruj-2023).

[27] T. Stackpole, 'What Is Web3?', *Harvard Business Review*, May 10, 2022. Pristupljeno: 04-ruj-2023. [Online]. Available: https://hbr.org/2022/05/what-is-web3