Pametni ugovori o alotmanu

Svetislav Simić*
* Fakultet tehničkih nauka, Univerzitet u Novom Sadu, Novi Sad, Srbija
simicsvetislav@uns.ac.rs

***Apstrakt** – Ovaj rad predstavlja teorijsko i praktično istraživanje o mogućnostima primene pametnih ugovora u domenu prava. Kao primer ugovora koji je predstavljen u vidu pametnog ugovora odabran je ugovor o alotmanu, to jest, ugovor kojim ugostitelj ustupa turističkoj agenciji određene smeštajne kapacitete, a koje agencija potom izdaje, u vidu aranžmana, krajnjim korisnicima. Prelazak sa tradicionalnog sklapanja ugovora na sklapanje pametnih ugovora bi značajno promenilo način poslovanja. Ugovori bi se na ovaj način sklapali bez posrednika. Samim tim ugovori bi mogli da se sklapaju na jednostavniji i efikasniji način, a troškovi vezani za ugovor bi bili smanjeni. U radu je opisana veb aplikacija, koja omogućava korisnicima da putem nje sklapaju ugovore i vrše potrebne interakcije sa njima, a pri tome se oslanjajući na blokčejn tehnologiju. Pametni ugovor je pisan korišćenjem Solidity programskog jezika, sa ciljem da se izvršava na Ethereum platformi. Takođe analizirane su slične primene blokčejn tehnologije i predstavljene su razne praktične i administrativne prepreke, zbog kojih primena pametnih ugovora na ovakav način nije u potpunosti moguća.*

Ključne reči: pametni ugovori, blokčejn, *Ethereum*, sklapanje ugovora, alotman

I. UVOD

Zadatak ovog rada je da se ispitaju mogućnosti primene pametnih ugovora, kao zamene za, pre svega, pisane, ali i usmene ugovore. Ugovorne strane sklapaju ugovor kako bi se pravno obavezale da će izvršiti svoje obaveze prema drugoj strani. Ugovor se smatra sklopljenim kada ugovorne strane potvrde da su saglasne sa predloženim ugovorom. Dok se usmeni ugovori sklapaju na osnovu usmenog dogovora strana, gde način interpretacije takvog ugovora može biti veoma širok, ugovori koji imaju veći značaj za strane i u sebi sadrže precizne odredbe obično se sklapaju pisanim putem.

Sklapanje ugovora u pisanoj formi zahteva od strana obrazovanje u oblasti prava, ili da imaju pomoć nekog pravnika. Pravnici bi trebalo da protumače da li u ugovoru piše ono što je i predviđeno, da li postoje propusti koji bi mogli da izazovu zloupotrebu ugovora jedne od strana, da li je ugovor u skladu sa pravnim aktima koji važe na nekoj teritoriji i slično. Za razliku od tradicionalnih ugovora, pametni ugovori se ne oslanjaju na treću stranu, odnosno nekog posrednika, prilikom svoje realizacije. Posledica ovoga bi trebala da bude jednostavnija realizacija ugovora, kao i rešavanje eventualnih sporova, pošto je izvršavanje ugovora algoritamski vođeno, kao i niži troškovi ugovornih strana, zbog eliminisanja posrednika.

U ovom radu će biti opisan sistem koji omogućava ugostiteljsko-turističkim organizacijama (u nastavku teksta ugostitelj) i agencijama, koji su prethodno registrovani kao korisnici sistema, da pregovoraju, vrše sklapanje i interakciju sa pametnim ugovorima o alotmanu, posredstvom veb aplikacije. Pomenuta veb aplikacija vrši interakciju sa blokčejn testnom mrežom na koju se smeštaju ugovori o kojima strane pregovaraju, a potom ih i sklapaju. Cilj upotrebe ovakvog sistema je da olakša i ubrza sklapanje ugovora i da smanji troškove ugovornih strana.

Ugovor o alotmanu je tip ugovora kod koga ugostitelj stavlja na raspolaganje turističkoj agenciji određenu količinu smeštajnih kapaciteta koje poseduje u nekom objektu, na ograničeni vremenski period. Dužnost turističke agencije je da pronađe lica zainteresovana za ustupljene smeštaje, a ugostitelja da pruži usluge licima koja se odluče za zakup smeštaja posredstvom agencije. Turistička agencija za svaki realizovani aranžman plaća ugostitelju definisanu cenu, a zauzvrat dobija odgovarajuću proviziju, dok kada nije u mogućnosti da izda smeštaj ima obavezu da o tome obavesti ugostitelja u određenom vremenskom roku pre nastupanja termina za koji smeštaj nije izdat. Izuzetak predstavljaju ugovori o alotmanu sa garancijom, na osnovu kojih je agencija dužna da popuni sve ustupljene kapacitete, a ukoliko to ne ostvari, dužna je da ugostitelju plati odgovarajuću naknadu, koja se računa na osnovu broja neiskorišćenih lažaja po danu, odnosno, nerealizovanih noćenja. Smeštajni kapaciteti koje je ugostitelj u nekom vremenskom periodu ugovorom već ustupio nekoj od turističkih agencija ne mogu biti predmet novog ugovora o alotmanu za vremenski period koji se potpuno ili delimično poklapa sa periodom definisanim u nekom od već zaključenih ugovora.

Na teritoriji Republike Srbije prava i obaveze ugovornih strana, koje su sklopile pravnoobavezujući ugovor, su uređena Zakonom o obligacionim odnosima (skraćeno ZOO) [1]. Zakon nalaže da se ugovor o alotmanu mora zaključiti u pisanoj formi. To podrazumeva da se ugovor sklapa uz postojanje odgovarajućih pravnih elemenata, koji daju pravnu snagu sklopljenom ugovoru.

Kritični deo ovog sistema je pametni ugovor, koji treba da na što bolji način predstavi ugovor o alotmanu. Suštinski, pametni ugovor je programski kod koji se nalazi i izvršava na blokčejnu. Ideju pametnih ugovora prvi je izneo Nick Szabo 1994. godine [2], [3]. Konkretno rešenje za koje se smatra da je omogućilo razvoj pametnih ugovora je razvoj *Ethereum* virtualne mašine (EVM) 2014. godine, koja pruža mogućnost izvršavanja Tjuring-kompletnih (*Turing complete*) programa na blokčejnu.

Treba napomenuti terminološku razliku između pametnih ugovora u domenu prava (*smart legal contracts*) i pametnih ugovora (*smart contracts*). Prvi termin se koristi kada se misli na programe napisane tako da reprezentuju pravnoobavezujuće ugovore koje sklapaju ugovorne strane, dok se pod drugim terminom smatra bilo koji programski kod koji je napisan sa ciljem da se *deploy*-uje, verifikuje i izvršava na nekoj blokčejn platformi [4].

Termin blokčejn (*blockchain*) je pomenut već više puta, pa će ovom prilikom i on biti pojašnjen. Blokčejn je sačinjen od liste zapisa, čiju identičnu kopiju sadrže svi čvorovina mreži. Čvorovi čine mrežu, u kojoj su, obično, ali ne nužno, svi čvorovi ravnopravni, odnosno ne postoji neki centralni autoritet pri donošenju odluka. Ovakva organizacija mreže se na engleskom naziva *distributed ledger* ili *distributed ledger technology* (DLT). Ukoliko se bilo ko može priključiti mreži kao čvor, govorimo o javnoj mreži, a u suprotnom o privatnoj.

Pomenuta lista zapisa se sastoji od blokova (*blocks*). Blokovi se međusobno uvezuju u lanac (*chain*), tako da svaki blok, po pravilu, ima *hash* vrednost prethodnog bloka, vremenski žig (*timestamp*) kreiranja bloka, redni broj bloka, svoju *hash* vrednost, podatke vezane za transakcije i *nonce* vrednost. Blok može sadržati podatke o više transakcija. Transakcija predstavlja digitalno potpisan paket podataka koji izvršava jednu interakciju sa blokčejnom. Prvi blok u lancu se naziva *genesis* blok i on jedini sme da ima nevalidnu *hash* vrednost prethodnog bloka.

Danas se za razvoj pametnih ugovora mogu koristiti različite platforme, ali najčešće se koristi, već pomenuta, *Ethereum* platforma [5]. *Ethereum* predstavlja ciljanu platformu za razvijeni sistem. Ona spada u drugu generaciju blokčejn platformi [6], koja, za razliku od prve generacije, koja se zasnivala na kriptovalutama, uvodi određena proširenja, tako da omogućava razvoj kompleksnih distribuiranih aplikacija. Osnovna kriptovaluta na ovoj platformi se naziva *ether*. Postoje manje i veće jedinice. Najmanja je *wei* (1 *ether* = 10^{18}*wei*), a najveća Tether (1 *ether* = 10^{-12}*Tether*). Takođe, *Ethereum* spada u, takozvane, javne blokčejn mreže. To znači da se bilo koji korisnik može anonimno pridružiti mreži, kreirati nove transakcije i/ili učestvovati u verifikaciji novih blokova.

Ethereum omogućava sastavljanje pametnih ugovora pomoću više programskih jezika visokog nivoa. Programski kod pametnih ugovora na mreži se izvršava u okviru *Ethereum* virtualne mašine (*Ethereum virtual machine* – EVM), a kako bi mogao da se ivršava prvo je neophodno da se prevede u odgovarajući *bytecode*.

Pri razvoju pametnih ugovora za *Ethereum* mrežu najčešće se koristi programski jezik *Solidity* [7], [8]. To je objektno-orijentisani programski jezik visokog nivoa za sastavljanje pametnih ugovora za *Ethereum* platformu.

Kraći pregled istraživanja koja su se bavila idejom pametnih ugovora, oblastima gde se oni mogu primeniti, kao i postojećih projekata koji se baziraju na upotrebi pametnih ugovora, dat je u drugom poglavlju. Treće poglavlje predstavlja sistem iz ugla krajnjih korisnika i opisuje način na koji oni mogu da vrše interakciju sa njim. Određeni implementacioni detalji, sa posebnim akcentom na razvoj i interakciju sa pametnim ugovorima, opisani su u četvrtom poglavlju. U petom poglavlju su razmotreni neki aspekti u realnoj primeni ovakvih ugovora i način na koji bi uticali na aktere, u odnosu na korišćenje tradicionalnog načina sklapanja ugovora. Konačno, u šestom poglavlju je još jednom ukratko opisan ceo rad, diskutovana su neka otvorena pitanja i predloženi su pravci za dalji razvoj implementiranog sistema.

II. SRODNA ISTRAŽIVANJA

Analizom literature nije se došlo do naučnih radova koji rešavaju konkretan problem ugovora o alotmanu, ali će biti predstavljeni radovi koji se bave raznim teorijskim i praktičnim temama, koje su vezane za pametne ugovore. Dodatno, biće opisani postojeći sistemi i platforme koji se baziraju na upotrebi pametnih ugovora.

U [5] autori su izdvojili naučne radove koji se bave temom pametnih ugovora i analizirali su kojim aspektima pametnih ugovora se oni bave. Kombinacijom automatskih i ručnih metoda filtriranja radova izdvojili su 24 rada. Dve trećine od tih radova se bave problemima pri razvoju pametnih ugovora. Izdvojene su četiri grupe problema, a to su problemi prilikom pisanja ispravnog programskog koda, problem bezbednosti, problem privatnosti i pitanje performansi. Ostatak izdvojenih radova se bavi primerima realne primene pametnih ugovora i teorijskim mogućnostima njihove primene. Kao primeri realne primene pojavljuju se primene pametnih ugovora u trgovini, poštenoj razmeni, Internetu stvari (*Internet of Things* – IoT) i kontroli pristupa.

U [9] analizirani su 48 naučnih radova, čije je zajedničko pitanje kako uspešno primeniti pametne ugovore u savremenoj kompaniji. Ustanovili su da se kompanije koje najviše prihvataju korišćenje pametnih ugovora bave menadžmentom lanaca snabdevanja, finansijama, zdravstvenom negom, bezbednošću informacija, pametnim gradovima i IoT rešenjima.

Da se pametni ugovori najviše koriste u oblasti finansija, a da im je najveći problem bezbednost, zaključak je u [10]. Takođe, kao prednosti pametnih ugovora izdvojeni su brzina, preciznost, manji rizik prilikom izvršavanja, niža cena i potencijal za pojavu novih poslovnih modela, a kao dodatni problemi skalabilnost, fleksibilnost i privatnost.

Platforma *Corda* je nastala sa ciljem stvaranja globalne mreže distribuiranih čvorova, koja vodi evidenciju o pravnim ugovorima, njihovom stanju, obavezama ugovornih strana i koja vrši procesiranje stavki takvih ugovora [11]. Razvoj ove platforme predvodi kompanija R3 [12], koja zajedno sa još 300 drugih kompanija radi na tome da omogući korišćenje ove platforme u sektorima kao što su finansijske usluge, osiguranje, zdravstvena nega, trgovina i digitalna imovina [13]. Pametni ugovori za ovu platformu se mogu pisati na bilo kom programskom jeziku koji se prevodi u *Java bytecode*, to jest, koji se piše sa ciljem izvršavanja na *Java* virtualnoj mašini (*Java virtual machine* – JVM).

Rentberry [14] je platforma za izdavanje i kupoprodaju zasnovana na tehnologiji blokčejna. Cilj platforme je da reši sve ozbiljne probleme koji danas postoje u tradicionalnom izdavanju i kupoprodaji nekretnina [8].

Projekat *Hyperledger* je krovni projekat za veći broj drugih projekata čiji je cilj da se razviju radni okviri, alati i biblioteke neophodni za razvoj sistema baziranih na blokčejnu, koji bi se koristili u velikim organizacijama i, samim tim, na globalnom nivou [15]. Za problem koji se rešava u ovom radu posebno je zanimljiv radni okvir *Hyperledger Fabric* [16]. Ovaj radni okvir predstavlja infrastrukturu za privatnu blokčejn mrežu i obezbeđuje njenu modularnu arhitekturu, koja se može menjati u zavisnosti od potreba korisnika. Takođe, omogućava izvršavanje pametnih ugovora, podelu uloga za čvorove u mreži i podesivi *consensus protocol* na mreži, a pri svemu tome vodeći računa o očuvanju privatnosti [17]. U ovom okruženju, za razvoj pametnih ugovora se mogu koristiti neki od standardnih jezika opšte namene, a to su *JavaSrcipt* [18], *Go* [19] i *Java* [20].

III. METOD

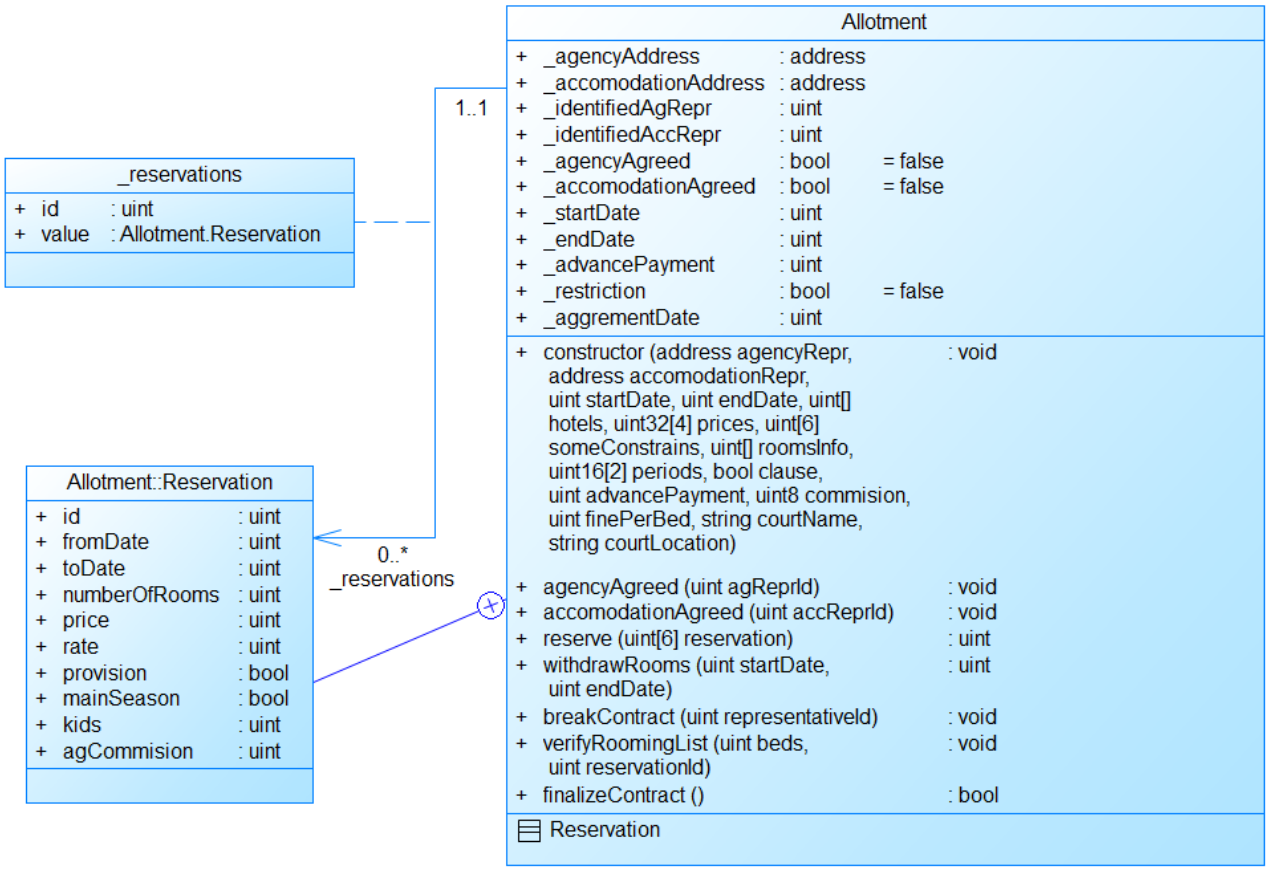
U ovom poglavlju će biti opisano kako je zamišljeno sklapanje pametnih ugovora, kao i interakcija koju korisnik ostvaruje sa sistemom prilikom njegovog korišćenja. Biće predstavljeno u kojim situacijama korisnik može da obavi koji vid interakcije i kako može izgledati životni ciklus jednog ugovora.

Finalni proizvod istraživanja koje je opisano u ovom radu je softversko rešenje, u vidu veb aplikacije, koje ima za cilj da omogući parametrizovanje, sklapanje i vršenje interakcije sa pametnim ugovorima o alotmanu. Ugovori koji bi se sklapili na ovakav način bi mogli da predstavljaju samostalne pravnoobavezujuće ugovore, ukoliko bi za to postojale mogućnosti, ili bi mogli da predstavljaju dopune za ugovore koji bi se sklapali na tradicionalan način, i u tom slučaju ne bi imali pravnu snagu, ali bi omogućili automatizovano izvršavanje stavki ugovora.

Pametni ugovor je razvijan sa ciljem da na što je moguće bolji način predstavi odredbe pravnog ugovora o alotmanu i da omogući izvršavanje njegovih odredbi. Kao uzor je korišćen model ugovora o alotmanu dostupan na [21]. Odredbe ugovora se utvrđuju u fazi pregovaranja i kada se uspostavi saglasnost volja između strana, odredbe ugovora više nije moguće menjati. Korisnici putem korisničkom interfejsa mogu vršiti određene interakcije sa ugovorom, s tim da se vodi računa o tome koji korisnik sme da izvrši koje operacije i koje podatke o ugovoru sme da vidi. Na slici 1 je prikazan blago uprošćeni dijagram klasa za pametni ugovor, na kojem je moguće videti koji podaci o ugovoru se čuvaju i koje vidove interakcije je moguće izvršiti.

Biće prikazana analiza kako se odnose neke od ključnih odredbi pravnog ugovora o alotmanu sa funkcionalnostima koje nudi pametni ugovor. Za početak, u prvom članu modela ugovora [21] se određuje u kom periodu, u kojim objektima i po kojim cenama ugostitelj ustupa turističkoj agenciji smeštajne kapacitete. Da bi ova, verovatno i ključna odredba, bila ispoštovana u pametnom ugovoru potrebno je da veći broj različitih metoda funkcioniše na očekivani način. Prilikom sklapanja ugovora je neophodno da se dostave svi potrebni podaci, koji moraju biti ispravni. Kada predstavnik agencije ugovori neki aranžman i prosledi ugovoru informacije o rezervaciji, tada se proverava njihova ispravnost, da li je datum u odgovarajućem opsegu, da li postoji dovoljno slobodnih kapaciteta. Tačna količina kapaciteta koji se ustupaju određena je članom šest modela ugovora. Takođe, ugovor mora biti u aktivnom stanju, za ugovore za koje ne postoji saglasnost volja ili su raskinuti ili istekli rezervacije ne mogu biti napravljene. Ukoliko rezervacija može da se realizuje, proverava se da li je period rezervacije u glavnoj sezoni, kako bi mogla da se odredi tačna cena, i nakon toga se podaci o njoj trajno čuvaju. Za svaku rezervaciju je neophodno izvršiti overu ruming lista, što je utvrđeno članom devet u modelu ugovora. U pametnom ugovoru postoji funkcija kojom se vrši overavanje određene ruming liste. U okviru iste funkcije izvršava se prenos odgovarajuće količine sredstava, u zavisnosti od cene overene rezervacije, sa naloga ugovora na nalog ugostitelja. Na taj način se ispunjava član deset modela ugovora.

Kako bi započeo korišćenje sistema korisnik mora da izvrši autentifikaciju unosom svog korisničkog imena i



Slika 1 - Uprošćeni dijagram klasa pametnog ugovora

šifre. Korsinik koji nije autentifikovan ne može izvršiti interakciju sa sistemom na bilo koji način, osim sa onim delom sistema preko koga se vrši autentifikacija.

Kada je korisnik potvrdio svoj identitet, unošenjem odgovarajućih kredencijala, može pristupiti vršenju određenih interakcija sa određenim ugovorima. Smatra se da je svaki korisnik predstavnik jedne agencije ili ugostitelja. S toga, on može imati uvid i vršiti interakciju samo sa ugovorima koji se odnose na agenciju ili ugostitelja koje predstavlja.

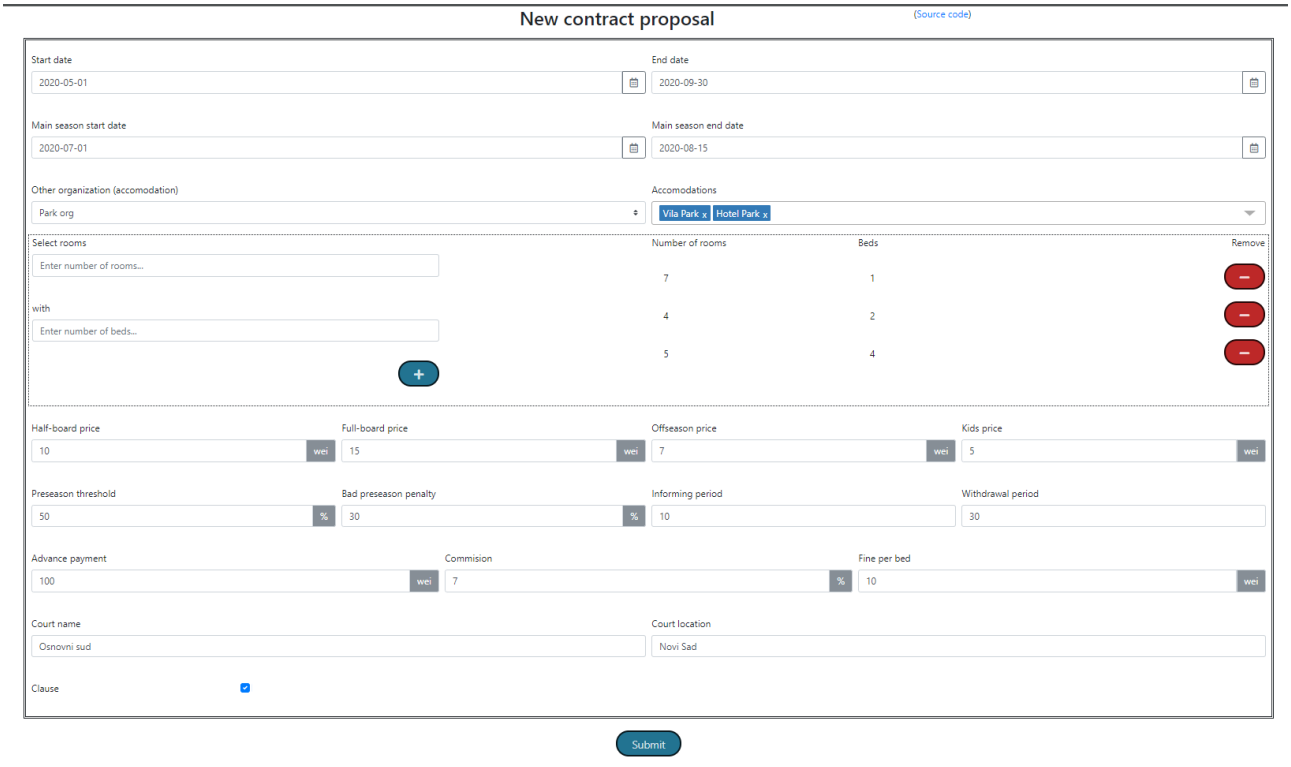
Prva faza kod sklapanja ugovora je faza pregovaranja. Predstavnici agencija mogu predstavnicima ugostitelja poslati predloge za sklapanje ugovora o alotmanu, ali važi i obrnuto, odnosno, predstavnici ugostitelja takođe mogu slati predloge predstavnicima agencija. U predlogu ugovora moraju biti navedeni svi podaci koji su neophodni za sklapanje ugovora. Na prispeli predlog može odgovoriti bilo ko od predstavnika organizacije kojoj je predlog bio upućen. Odgovor može biti potvrdan, odričan ili u vidu kontra-ponude. Samo treći oblik odgovora produžava fazu pregovaranja i sastoji se od modifikovane ponude, koja se šalje nazad organizaciji koja je poslala prvu ponudu. Faza pregovaranja se na ovaj način može produžavati dokle god neka od organizacija ne prihvati ili odbije trenutnu ponudu.

Obe ugovorne strane su dužne da pre sklapanja ugovora daju novčano pokriće, za slučaj nepoštovanja ugovora. Kada postoji saglasnost volja i položena su pokrića, smatra se da je došlo do sklapanja ugovora. Za interakciju sa ugovorom su zaduženi predstavnici agencije i ugostitelja koji su predložili, odnosno prihvatili, uslove tokom pregovaranja.

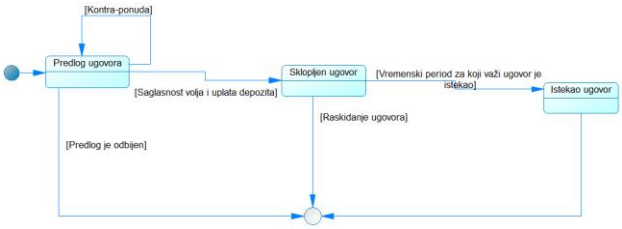
Odgovorni predstavnik agencije može vršiti različite interakcije sa ugovorom. Tu spadaju:

- rezervisanje određenih smeštajnih kapaciteta za određeni period;
- vraćanje nerezervisanih smeštajnih kapaciteta na korišćenje ugostitelju, ukoliko je to dozvoljeno;
- potvrđivanje ruming lista (ovu operaciju može izvršiti bilo koji predstavnik agencije).

Sve relevantne informacije vezane za sklopljene ugovore ili ugovore o kojima se pregovara mogu videti svi predstavnici odgovarajućih agencija, s tim da ne mogu sa njima vršiti proizvoljne vidove interakcije, kao što je već i opisano. Odredbe sklopljenih ugovora se ni na koji način ne mogu menjati nakon što je izvršeno sklapanje ugovora.



Slika 2 - Forma koja se popunjava pri slanju predloga ugovora



Slika 3 - Dijagram stanja životnog ciklusa pametnog ugovora

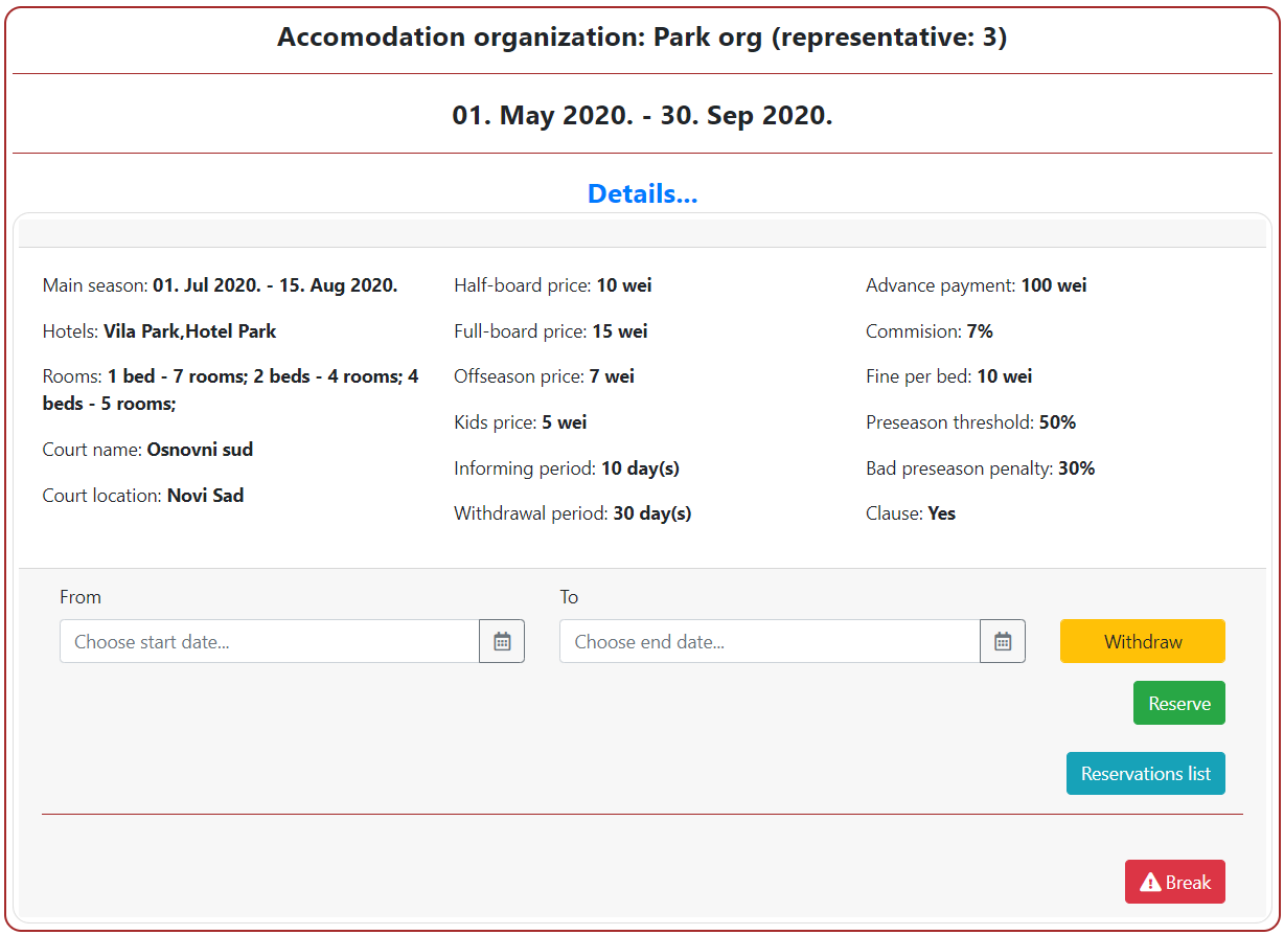
Životni ciklus sklopljenog ugovora se može završiti na dva načina. Jedan način je da protekne čitav ugovorni period. Tada se ugovor smatra isteklim i proveravaju se određene odredbe ugovora, kako bi se utvrdilo da li postoji osnova za obeštećenje neke od strana. Druga mogućnost je prevremeni raskid ugovora. Obe strane imaju određeni vremenski period u kome mogu raskinuti ugovor bez posledica. Međutim, ako taj period istekne, a neka od strana želi da raskine ugovor, tada ona mora drugoj strani uplatiti određenu odštetu, a ta suma se obezbeđuje od pokrića koje je uplaćeno pre sklapanja ugovora. Na slici 3 prikazan je dijagram stanja životnog ciklusa pametnog ugovora u implementiranom sistemu.

IV. Implementacija sistema

Implementirani sistem [22] je višeslojna veb aplikacija, u kojoj se izdvajaju prezentacioni sloj, aplikacioni sloj, sloj pristupa podacima i sloj pametnih ugovora.

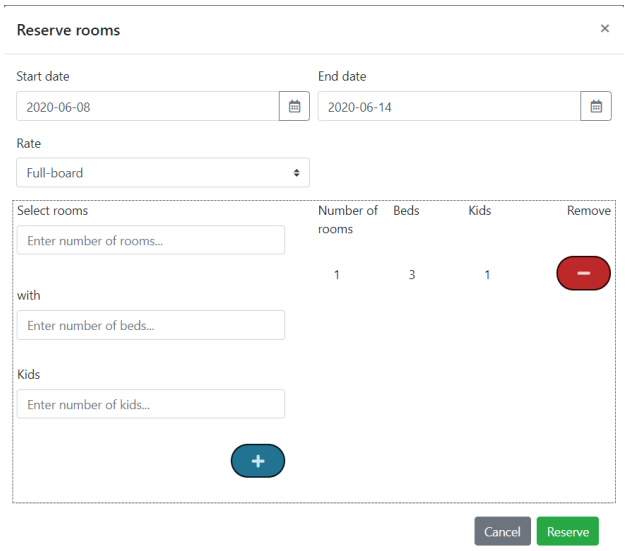
Prezentacioni sloj, ili kraće *frontend*, je nezavisna aplikacija implementirana korišćenjem *Angular* radnog okvira [23], koja omogućava korisnicima interakciju sa sistemom. Pomoću nje korisnici mogu da pregovaraju o sklapanju ugovora, sklapaju ugovore, pregledaju detalje svojih sklopljenih ugovora i vrše dozvoljene interakcije sa ugovorima. Tako predstavnici turističkih agencija mogu rezervisati smeštaj koji su izdali gostima, vršiti overu ruming lista, vratiti smeštajne kapacitete na korišćenje ugostitelju, ukoliko nisu uspeli da ih izdaju i ukoliko je ugovorom to dozvoljeno, a obe strane mogu zahtevati raskid ugovora.

Na slici 2 prikazan je izgled popunjene forme koju korisnici popunjavaju prilikom slanja predloga ugovora. Forma izgleda identično i kada se šalje kontra-ponuda. Ponašanje forme se za nijansu razlikuje u zavisnosti od toga da li predlog popunjava predstavnik agencije ili ugostitelja. Razlika se odnosi na polje za odabir smeštajnih objekata u kojima se ustupaju smeštajni kapaciteti (polje *Accomodations*). Kada predstavnik agencije popunjava formu, dozvoljene vrednosti zavise od vrednosti u polju u kojem se bira ugostitelj kome se ponuda šalje (*Other organization*), to jest, moguće je odabrati samo one objekte koji su u vlasništvu odabranog ugostitelja. Kada formu popunjava ugostitelj, za polje *Accomodations* je moguće odabrati samo one kapacitete koje taj ugostitelj poseduje.



Slika 4 - Informacije o ugovoru i moguće akcije

Na slici 4 prikazan je deo interfejsa koji ovlašćenom predstavniku agencije predstavlja podatke o sklopljenom ugovoru i nudi mu mogućnost izvršavanja određenih operacija. Klikom na dugme sa natpisom *Reserve* otvara se dijalog u kome je nakon unosa potrebnih podataka moguće rezervisati željene kapacitete, dok je listu rezervacija, u okviru koje je moguće izvršiti i verifikaciju ruming lista, moguće dobiti klikom na dugme sa natpisom



Slika 5 - Forma za dodavanje nove rezervacije

Reservations list. Izgled tog dijaloga je prikazan na slici 5.

Serverska ili *backend* aplikacija je implementirana korišćenjem programskog jezika *Java* [20] i *Spring* [24] radnog okvira, preciznije, njegove *Spring Boot* [25] verzije. Ova aplikacija objedinjuje nekoliko slojeva. To je pre svega aplikacioni sloj, koji sadrži REST *endpoint*-e kojima se uspostavlja komunikacija sa *frontend* aplikacijom, a pored toga postoji sloj za pristup podacima, odnosno komunikaciju sa MySQL sistemom za upravljanje bazama podataka [26].

Može se reći da u serverskoj aplikaciji postoji još jedan sloj, a to je sloj za interakciju sa pametnim ugovorima i o tome će biti nešto više reči. Pomenutu interakciju omogućava *web3j* biblioteka [27]. To je biblioteka napisana za *Java* programski jezik, koja, između ostalog, omogućava postavljanje pametnih ugovora na mrežu, pozivanje funkcija pametnih ugovora i transfer kriptovalute sa jednog na drugi račun na mreži. Ova biblioteka pruža fleksibilnost koja je dovoljna za sve potrebne interakcije sa blokčejnom.

Sloj od najvećeg značaja za ovaj sistem je sloj pametnih ugovora. Kao što je već napomenuto, ugovori su pisani za *Ethereum* kao ciljanu platformu i korišćen je *Solidity* programski jezik, i to verzija 0.6. Treba imati u vidu da je *Solidity* jezik koji je u intenzivnom razvoju, pa trenutno ne obezbeđuje sve pogodnosti koje su nezaobilazne u nekim drugim modernim programskim jezicima. Tako nije moguće korisiti brojeve sa decimalnim zarezom, već se željene operacije sa takvim brojevima moraju simulirati na neki drugi način. Takođe, ne postoji potpuna podrška za rad sa nizovima i svaka funkcija ima ograničenu veličinu steka, koja je prilično skromna, a to znači da nije moguće imati proizvoljan broj lokalnih promenljivih ili parametara funkcije.

Za razvoj pametnih ugovora korišćeno je veb-bazirano integrisano razvojno okruženje (*integrated development environment* – IDE) *Remix* [28]. Ovo okruženje omogućava razvoj pametnih ugovora za *Ethereum* platformu, sa podrškom za kompajliranje, *deployment*, testiranje i debagovanje ugovora. Razvojno okruženje ne zahteva postojanje nikakvog dodatnog softvera na korišćenom računaru, jedino je neophodno postojanje veb *browser*-a, pomoću koga se pristupa okruženju.

Ugovor o alotmanu je opisan pomoću jednog pametnog ugovora [29]. Napisani ugovor koristi funkcionalnosti nekih eksternih biblioteka, koje su takođe napisane u jeziku *Solidity*, tačnije njih četiri. Tri biblioteke su javno dostupne, a jedna je dodatno implementirana za ovaj sistem. Te biblioteke su:

- *SafeMath* [30] - matematičke operacije koje potencijalno mogu izazvati *overflow* u programskom jeziku *Solidity* ne izazivaju grešku prilikom izvršavanja i zbog toga njihovo korišćenje može izazvati neočekivano ponašanje. Ova biblioteka rešava taj problem tako što postavlja određene zahteve prilikom izvršavanja osnovnih operacija i u slučaju da se dogodi *overflow*, prijavljuje da je došlo do izuzetka;
- *strings* [31] - biblioteka koja obezbeđuje implementaciju različitih korisnih funkcija za rad sa stringovima, odnosno nizovima karaktera. Konkretno, korišćena je funkcija za konkatenaciju stringova;
- *BokkyPooBahsDateTimeLibrary* [32] - biblioteka koja omogućuje manipulisanje i preračunavanje vremenskih i datumskih vrednosti. Značaj ove biblioteke za sistem proizilazi iz toga što programski jezik *Solidity* nema tip podataka za predstavljanje datuma i zbog toga su datumi čuvani kao *timestamp*-ovi, koji predstavljaju vremenski trenutak u datumu koji se želi predstaviti, konkretno, podne po Griničkom srednjem vremenu (*Greenwich mean time* – GMT);
- *DateUtilsLibrary* – dodatno implementirana biblioteka, koja koristi funkcionalnosti *BokkyPooBahsDateTimeLibrary* kako bi se proveravali odnosi datuma i datumskih intervala, za situacije koje treba proveriti u pametnom ugovoru. Tako, postoje funkcije za poređenje datuma, proveravanje da li dva intervala imaju preklapanje, proveravanje da li jedan, širi interval obuhvata kompletan drugi interval, da li se datum nalazi unutar nekog intervala i da li je trenutni datum unutar zadate udaljenosti od nekog drugog datuma.

Sistem koji je rezultat ovog rada je u suštini centralizovan i pokušava da ograniči pristup pametnom ugovoru koji se nalazi na mreži. Centralizovan je zbog toga što se sve transakcije iniciraju iz *backend* aplikacije, a pristup se ograničava različitim metodama autentifikacije i autorizacije, na svim slojevima sistema. Konkretno, za kontrolu pristupa *backend* aplikaciji korišćen je *Spring Security* [34] radni okvir i JSON veb tokeni (*JSON web tokens* – JWT) [35], kao mehanizam za potvrdu prava korisnika da smeju da izvrše određene radnje.

Na nivou pametnih ugovora postoje provere koje dozvoljavaju samo predstavnicima agencije ili ugostitelja

da obave određene operacije. Te provere su zasnovane na mehanizmu modifikatora i provere broja naloga sa kojeg je inicirana transakcija, a da bi transakcija bila uopšte pokrenuta preko nekog naloga, nephodno je priložiti i privatni ključ, koji je povezan sa tim nalogom, a pretpostavka je da taj podatak može imati samo odgovarajući korisnik. U principu, pošto je *Ethereum* javna mreža, niko ne može zabraniti korisnicima koji mogu da potvrde svoj identitet da iniciraju transakciju i mimo *backend* aplikacije, što narušava pretpostavljenu centralizovanost, ali kontrola pristupa mreži se i dalje proverava na identičan način.

V. DISKUSIJA

Testiranje ugovora u početnoj fazi razvoja je vršeno u samom *Remix* razvojnom okruženju. Zbog problema sa fleksibilnošću i potrebe za stalnim, repetitivnim ponavljanjem istih operacija, ovaj način testiranja je ubrzo napušten.

Testiranje rešenja tokom razvoja je pretežno vršeno korišćenjem alata *Postman* [36]. *Postman* je alat za integraciono testiranje API-ja, gde spadaju i REST *endpoint*-i, koji postoje u *backend* aplikaciji. On omogućava ponovljive i pouzdane testove, koji se mogu automatizovati. Testiranje REST servisa se vrši slanjem HTTP zahteva, u okviru kojih se mogu parametrizovati svi aspekti tih zahteva, u koje spadaju i zaglavlja, sadržaj tela i podaci vezani za autorizaciju. S toga, *endpoint*-i koji se koriste pri ovakvom testiranju mogu u neizmenjenom obliku da se koriste i u stvarnom sistemu.

Do sada nisu vršena formalna testiranja pametnog ugovora i čitavog sistema. U slučaju pokušaja da se od ovog sistema napravi komercijalni proizvod, upravo je formalno testiranje stvar na koju treba obratiti posebnu pažnju. Programski kod pametnog ugovora koji je jednom *deploy*-ovan na mrežu ne može naknadno biti promenjen. To znači da se u slučaju otkrivanja greške u funkcionisanju ugovora ne može jednostavno izvršiti ispravka te greške. Postoje predlozi kojim metodama možemo formalno utvrditi da je pametni ugovor napisan na ispravan način [37], [38], kao i kako postojeći ugovori mogu biti promenjeni ili poništeni na univerzalni način [39].

U ovom radu je značajna pažnja pripisana *Ethereum* platformi za razvoj pametnih ugovora, ali ona, iako jeste ciljana platforma, nije direktno korišćena. Korišćenje glavne, javne *Ethereum* mreže (*main network – Mainnet*) zahtevalo bi potrošnju *ether*-a koji ima stvarnu materijalnu vrednost, odnosno bilo bi potrebno uložiti novčana sredstva da bi se pametni ugovor uvezao u lanac blokova i da bi se vršile transakcije. Zbog toga je korišćena testna mreža (*testnet*) *Ganache* [40] (novija verzija *TestRPC* mreže [41]), koja je napravljena tako da imitira ponašanje glavne mreže, s tim da ne radi sa pravim novcem, i služi baš za testiranje pametnih ugovora. Od glavne mreže *Ganache* se razlikuje i po tome što je to privatna mreža. Takođe, rudarenje transakcija se, po podrazumevanim opcijama, izvršava momentalno [42], što dodatno olakšava testiranje.

Ukratko će biti prokomentarisani očekivani troškovi za određene transakcije tokom životnog ciklusa pametnog ugovora. *Gas* je mehanizam za određivanje cene transakcije. Svaka transakcija ima određenu cenu *gas*-a (*gas price*) i maksimalnu količinu *gas*-a koju transakcija može da potroši (*gas limit*). Cena određuje koliko *ether*-a će dobiti rudar koji pronađe tačnu *nonce* vrednost i srazmerna je količini potrošenog gasa po definisanoj ceni. Ove vrednosti određuje kreator transakcije. Potrošena količina *gas*-a pokazuje koliko je transakcija bila zahtevna, a ukoliko ga ne bude dovoljno, odnosno prekorači se limit, transakcija neće biti realizovana. U tabeli 1 su prikazane cene u dolarima za određene transakcije. Može se reći da su prikazane cene manje od onih koje bi morale da budu plaćene pravnom posredniku prilikom sklapanja ugovora i realizacije ugovora, tako da postoji finansijska isplativost ugovornih strana prilikom sklapanja pametnih ugovora u odnosu na sklapanje ugovora na tradicionalni način. Takođe, treba imati u vidu da bi se optimizacijom programskog koda moglo doći i do još nižih troškova pri interakcijama sa ugovorom.

Tabela 1 - Cene u dolarima za obavljanje tipičnih transakcija za različite cene gasa Izvor [43]

	Jeftino/Sporo (1.1 Gwei)	Umereno (3 Gwei)	Skupo/Brzo (10 Gwei)
Deploy	$0.659	$1.797	$5.990
Rezervacija	$0.037	$0.100	$0.333
Odustajanje za neki period	$0.014	$0.037	$0.123

Značajna prednost kod elektronskog sastavljanja ugovora preko opisanog, ili sličnog, sistema predstavlja i brzina sklapanja ugovora. Ceo proces pregovaranja bi mogao da bude izuzetno brzo završen, s obzirom na to da se neophodna interakcija može u potpunosti obaviti preko veb aplikacije. Čak i da na samom početku pregovaranja strane nisu saglasne po svim pitanjima, one mogu efikasno razmeniti ponude i doći do odredbi ugovora zadovoljavajućih za obe strane.

Iako eksploatacija pametnih ugovora obično podrazumeva da su posrednici suvišni, kod sklapanja pametnih ugovora u domenu prava oni ne mogu biti u potpunosti ignorisani. Programski kod pametnih ugovora bi svakako trebao da bude napisan tako da predvidi što veći broj potencijalno spornih situacija i obezbedi mehanizme da se one razreše na fer način. Međutim, postoje situacije u stvarnom svetu na koje programski kod ne može da utiče i zbog toga bi se u nekim situacijama ugovorne strane morale osloniti na pravni sistem prilikom razrešavanja sporova. Korišćenje pametnih ugovora u domenu prava bi moralo da ima jasno definisan pravni okvir u kome deluje, tako da državne institucije predstavljaju nezaobilazni činilac realne primene pametnih ugovora, koje bi im i davale pravni legitimitet.

Kada strane sklapaju ugovor od suštinske važnosti je da one jasno znaju oko čega su se sporazumele i koje su njihove obaveze koje proizilaze iz ugovora. Prilikom sklapanja ugovora na tradicionalan način za takvo tumačenje bi trebalo da bude zaduženo neko lice koje se razume u pravo. Kod pametnih ugovora situacija je malo komplikovanija. Na samom blokčejnu ugovori se nalaze prevedeni u *bytecode*. Ugovori u ovakvom obliku nisu pogodni za direktno tumačenje. Verzija ugovora koja bi mogla da se tumači od strane stručnih lica, pre svega programera, je izvorni kod ugovora. Zbog toga je bitno da korisnici imaju na raspolaganju mogućnost da pristupe sadržaju izvornog koda, ali, takođe, i da se uvere da

izvorni kod ugovora odgovara *bytecode*-u, koji se nalazi na blokčejnu. Da bi se izvršila takva provera može se koristiti neki od alata za verifikaciju pametnih ugovora, koji ima takvu mogućnost, a jedan takav alat je dostupan na [44]. Na ovaj način se, praktično, programeri uvode kao novi posrednici u čitav proces sklapanja pametnih ugovora. S obzirom da je jedan od osnovnih ciljeva upotrebe pametnih ugovora bio eliminisanje posrednika, ovo može predstavljati ozbiljan argument protiv upotrebe pametnih ugovora.

Otvoreno pitanje kod korišćenja pametnih ugovora predstavlja pitanje bezbednosti. Napadači pokušavaju da zloupotrebe propuste u pametnim ugovorima, ali i na niskom nivou, iskorišćavajući ranjivosti *Ethereum* virtualne mašine [45]. Žrtva jedne takve zloupotrebe 2016. godine bila je upravo *Ethereum* glavna mreža. Tada je ukraden ether u vrednosti od 50 miliona američkih dolara. Ova krađa je sanirana podelom mreže na dve nove mreže, pri čemu je jedna originalna mreža na kojoj se krađa dogodila, a novonastala mreža je imala izmenjeni protokol i poništene su transakcije koje su dovele do krađe.

VI. Zaključak

U ovom radu je razmatrana potencijalna primena pametnih ugovora, za zaključivanje ugovora u pravnom smislu, na nekoj blokčejn platofrmi. Predstavljen je sistem čiji je cilj da omogući sklapanje pametnih pravnih ugovora o alotmanu. Sistem je implementiran kao veb aplikacija. Pametni ugovor je napisan za *Ethereum*, kao ciljanu platformu.

Implementirani sistem može predstavljati osnovu za dalji rad i stvaranje kvalitetnijeg rešenja. Iako je sistem razvijen sa ciljem da predstavlja zaokruženo rešenje za postavljeni problem, izazovi koje treba rešiti da jedna ovakva aplikacija postane komercijalni proizvod su brojni. Da bi se to desilo na prvom mestu bi trebao da se pronađe odgovarajući mehanizam formalnog testiranja, na osnovu koga bi mogao da se garantuje njegov kvalitet. Potom bi trebalo da se dodatno poradi na bezbednosti sistema. Za to bi mogao da se upotrebi alat *OYENTE* [46] za koji autori rada [47] ističu da je u stanju da u pametnim ugovorima pronađe potencijalne bezbednosne propuste.

Postoje brojna druga otvorena pitanja kada je u pitanju razvoj pametnih ugovora u oblasti prava. Neki autori tvrde da proceduralni način njihovog zapisivanja nije prikladan i da bi logičnije bilo da se oni zapisuju nekim deklarativnim jezikom [48], [49]. Bezbednosni rizici su stalna tema. Sajber kriminal svetsku ekonomiju godišnje košta oko milijardu dolara [50].

Još jedan značajan problem praktične prirode je to što pametni ugovori nisu regulisani zakonskom regulativom praktično nigde u svetu. U različitim državama odnos prema njima je prilično neharmonizovan i odnos prema njima zavisi od toga kako se oni tumače na osnovu postojećih regulativa [50]. U Evropskoj Uniji postoji dodatni problem, jer nedavno doneti GDPR (*General Data Protection Regulation*), koji treba da obezbedi pravni okvir za oblasti zaštite podataka i privatnosti na teritoriji Evropske Unije, ima stavke koje su konfliktne sa samim osnovama na kojima počiva blokčejn tehnologija, poput „prava da neko bude zaboravljen“ i prava na zaštitu od odluka koje su nastale isključivo na osnovu automatske obrade podataka [51].

Ovde su navedeni samo neki od problema koji postoje sa realnom upotrebom pametnih ugovora u oblasti prava. Takvih problema očito nije malo, ali se, sa stanovišta informatičara, intezivno radi na njima. Situacija pokazuje da trenutno nema mnogo prostora za njihovu realnu i samostalnu upotrebu, ali treba imati u vidu da je blokčejn relativno nova tehnologija, koja se neprestano razvija, i da se njen potencijlani disruptivni uticaj u poslovanju tek može očekivati.

Literatura

[1] Zakon o obligacionim odnosima – http://www.pravno-informacioni-sistem.rs/SlGlasnikPortal/eli/rep/slsfrj/skupstina/zakon/1978/29/1/reg (posećeno 15. aprila 2020.)

[2] SZABO, Nick. Smart contracts. Unpublished manuscript, 1994.

[3] SZABO, Nick. Formalizing and securing relationships on public networks. First Monday, 1997, 2.9.

[4] https://www.coindesk.com/making-sense-smart-contracts (posećeno 15. aprila 2020.)

[5] ALHARBY, Maher; VAN MOORSEL, Aad. Blockchain-based smart contracts: A systematic mapping study. arXiv preprint arXiv:1710.06372, 2017.

[6] XU, Xiwei, et al. The blockchain as a software connector. In: 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA). IEEE, 2016. p. 182-191.

[7] Solidity – programming language, 2020. https://solidity.readthedocs.io (posećeno 15. aprila 2020.)

[8] MOHANTY, Debajani. Ethereum for Architects and Developers. Apress, 2018. p. 35.

[9] UDOKWU, Chibuzor, et al. The state of the art for blockchain-enabled smart-contract applications in the organization. In: 2018 Ivannikov Ispras Open Conference (ISPRAS). IEEE, 2018. p. 137-144.

[10] MOHANTA, Bhabendu Kumar; PANDA, Soumyashree S.; JENA, Debasish. An overview of smart contract and use cases in blockchain technology. In: 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2018. p. 1-4.

[11] BROWN, Richard Gendal. The corda platform: An introduction. Retrieved, 2018, 27: 2018.

[12] R3 – DLT & Blockchain Software Development Company, 2020. https://www.r3.com (posećeno 15. aprila 2020.)

[13] https://www.fnlondon.com/articles/blockchain-initiative-of-the-year-2018-the-nominees-20180325 (posećeno 15. aprila2020.)

[14] Rentberry –Apartments for Rent Worldwide, 2020. https://rentberry.com (posećeno15. aprila 2020.)

[15] Hyperledger – Open Source Blockchain Technologies, 2020. https://www.hyperledger.org/ (posećeno 15. aprila 2020.)

[16] Hyperledger Fabric – Distributed ledger software, 2020. https://www.hyperledger.org/projects/fabric (posećeno 15. aprila 2020.)

[17] ANDROULAKI, Elli, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference. 2018. p. 1-15.

[18] JavaScript – programming language, 2020. https://www.javascript.com/ (posećeno 15. aprila 2020.)

[19] Go – programming language, 2020. https://golang.org/ (posećeno 15. aprila 2020.)

[20] Java – programming language, 2020. https://www.java.com/ (posećeno 15. aprila 2020.)

[21] Model ugovora o alotmanu – http://www.biroaura.co.rs/pdf/ostalo/ugovori/ugovor_o_alotmanu.pdf (posećeno 17. aprila 2020.)

[22] Github repozitorijum implementiranog rešenja – https://github.com/SimicSvetislav/allotment-smart-contracts (posećeno 16. aprila 2020.)

[23] Angular – platform for building mobile and desktop web applications, 2020. https://angular.io/ (posećeno 15. aprila 2020.)

[24] Spring – opensource Java-based framework, 2020. https://spring.io/projects/spring-framework (posećeno 15. aprila 2020.)

[25] Spring Boot – opensource Java-based framework, 2020. https://spring.io/projects/spring-boot (posećeno15. aprila 2020.)

[26] MySQL – relational database management system, 2020. https://www.mysql.com/ (posećeno 15. aprila 2020.)

[27] Web3j – Java and Android library for working with Smart Contracts and integrating with clients (nodes) on the Ethereum network, 2020. https://www.web3labs.com/web3j (posećeno15. aprila 2020.)

[28] Remix – web-based Ethereum IDE, 2020. https://remix.ethereum.org/ (posećeno 16. aprila 2020.)

[29] Izvorni kod pametnog ugovora – https://github.com/SimicSvetislav/allotment-smart-contracts/blob/master/contracts/Allotment.sol (posećeno 16. aprila 2020.)

[30] Safe Math – library for Solidity, 2020. https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/SafeMath.sol (posećeno 15. aprila 2020.)

[31] Strings – string utility library for Solidity, 2020. https://github.com/Arachnid/solidity-stringutils/blob/master/src/strings.sol (posećeno 15. aprila 2020.)

[32] BokkyPooBah's DateTimeLibrary – A gas-efficient Solidity date and time library, 2020. https://github.com/bokkypoobah/BokkyPooBahsDateTimeLibrary/blob/master/contracts/BokkyPooBahsDateTimeLibrary.sol posećeno 15. aprila 2020.)

[33] DateUtilsLibrary – utility library for calculations with dates, 2020. https://github.com/SimicSvetislav/allotment-smart-contracts/blob/master/contracts/DateUtilsLibrary.sol (posećeno 16. aprila 2020.)

[34] Spring security – Authentication and access-control framework, 2020. https://spring.io/projects/spring-security (posećeno 15. aprila 2020.)

[35] JSON web tokens – RFC 7519 method for representing claims securely between two parties, 2020. https://jwt.io/ (posećeno 15. aprila 2020.)

[36] Postman – The collaboration platform for API development, 2020. https://www.postman.com/ (posećeno 15. aprila 2020.)

[37] BHARGAVAN, Karthikeyan, et al. Formal verification of smart contracts: Short paper. In: Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security. 2016. p. 91-96.

[38] MAGAZZENI, Daniele; MCBURNEY, Peter; NASH, William. Validation and verification of smart contracts: A research agenda. Computer, 2017, 50.9: 50-57.

[39] MARINO, Bill; JUELS, Ari. Setting standards for altering and undoing smart contracts. In: International Symposium on Rules and Rule Markup Languages for the Semantic Web. Springer, Cham, 2016. p. 151-166

[40] Ganache CLI - A fast and customizable blockchain emulator, 2020. https://www.trufflesuite.com/docs/ganache/overview (posećeno 15. aprila 2020.) .

[41] https://www.trufflesuite.com/blog/testrpc-is-now-ganache (posećeno 15. aprila 2020.)

[42] https://github.com/trufflesuite/ganache-cli/#user-content-options (posećeno 15. aprila 2020.)

[43] https://ethgasstation.info/calculatorTxV.php (posećeno 15. aprila 2020.)

[44] https://etherscan.io/verifyContract (posećeno 16. aprila 2020.)

[45] ATZEI, Nicola; BARTOLETTI, Massimo; CIMOLI, Tiziana. A survey of attacks on Ethereum smart contracts. IACR Cryptology ePrint archive, 2016, 2016: 1007.

[46] OYENTE – An Analysis Tool for Smart Contracts, 2020. https://github.com/melonproject/oyente (posećeno 15. aprila 2020.)

[47] LUU, Loi, et al. Making smart contracts smarter. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016. p. 254-269.

[48] GOVERNATORI, Guido, et al. On legal contracts, imperative and declarative smart contracts, and blockchain systems. Artificial Intelligence and Law, 2018, 26.4: 377-409.

[49] IDELBERGER, Florian, et al. Evaluation of logic-based smart contracts for blockchain systems. In: International Symposium on Rules and Rule Markup Languages for the Semantic Web. Springer, Cham, 2016. p. 167-183.

[50] GIANCASPRO, Mark. Is a 'smart contract'really a smart idea? Insights from a legal perspective. Computer law & security review, 2017, 33.6: 825-835.

[51] https://www.eublockchainforum.eu/sites/default/files/reports/report_legal_v1.0.pdf?width=1024&height=800&iframe=true (posećeno 15. aprila 2020.)