PRIMJENA BLOCKCHAIN TEHNOLOGIJE U PREHRAMBENOJ INDUSTRIJI

Škegro, Andrej

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

PRIMJENA BLOCKCHAIN TEHNOLOGIJE U PREHRAMBENOJ INDUSTRIJI

Andrej Škegro

Zagreb, veljača 2019.

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada".

U Zagrebu, 26.2.2019.

Andrej Škegro

Predgovor

Zahvaljujem se svom mentoru profesoru Danielu Kučaku na strpljenju i pomoći pri izradi ovog završnog rada. Također želim se zahvaliti svim profesorima i asistentima Visokog učilišta „Algebra“ na svom prenesenom znanju koje su me učinili stručnim i kompetentnim na tržištu rada. Želim se zahvaliti svojoj djevojci Anamariji na velikoj pomoći, razumijevanju i vjeri koju je imala u mene tijekom studija. Hvala mojim roditeljima koji su me ohrabrivali i s nestrpljenjem čekali svaki rezultat ispita.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

Blockchain tehnologija jedna je od najatraktivnijih tehnologija u računalnom svijetu unazad deset godina. Pozornost je privukla 2008. godine kada je netko pod pseudonimom Satoshi Nakamoto iskoristio sve prednosti blockchain tehnologija za kreiranje digitalnog novca Bitcoin. Temeljne karakteristike *blockchain* tehnologije su nepromjenjivost i sigurnost. Jednom zapisani podaci ne mogu biti promijenjeni niti brisani, dok sigurnost pružaju kriptografski algoritmi. Ova tehnologija može se koristiti u raznim područjima kao što su financijski sektor, medicina, prehrambena industrija, trgovina i dr. U ovom radu detaljno je prikazano kreiranje programskog rješenja koristeći *blockchain* tehnologiju. Programsko rješenje omogućuje zapisivanje temperature izmjerene u nekom od četiri entiteta u lancu dostave hrane na *blockchain*. Takvim rješenjem rješava se problem povjerenja između entiteta, jer jednom zapisani podaci ne mogu biti izmijenjeni.

Ključne riječi: blockchain, sigurnost, kriptografija, Bitcoin.

Abstract

Blockchain technology is one of the most advanced technologies in the computer world back in ten years. It attracted attention in 2008 when someone under the pseudonym Satoshi Nakamoto exploited all the advantages of blockchain technology for creating digital money Bitcoin. The fundamental features of blockchain technology are immutability and security. Once written data can not be changed or deleted, while security provides cryptographic algorithms. This technology can be used in a variety of areas, such as the financial sector, medicine, food industry, commerce, etc. In this paper, we have detailed the creation of a software solution using blockchain technology. The program solution allows you to record the temperature measured in any of the four entities in the food chain at the blockchain. Such a solution resolves the problem of trust between the entities, because once recorded data can not be altered.

Ključne riječi: blockchain, security, cryptographic, Bitcoin.

Sadržaj

1. Uvod

Blockchain tehnologija omogućava prijenos digitalnih informacija i digitalnog novca kroz sve čvorove u lancu, gotovo bez mogućnosti kompromitiranja podataka. U lancu blokovi su međusobno povezani, a svaki blok može sadržavati jedan ili više zapisa. Blokovi se povezuju algoritmom koji koristi kriptografsku *hash* funkciju. Svaki čvor posjeduje svoj primjerak zapisa svih transakcija u lancu i nema potrebe za središnjim tijelom koji odobrava transakcije. Svaka transakcija se distribuirano kontrolira, te neispravna transakcija nikada ne može završiti u lancu. Ova tehnologija nastala je 2008. godine, njen osnivač je Satoshi Nakamoto, za potrebe distribuiranja digitalnog novca Bitcoin.

Postoje dvije vrste *blockchaina*, privatni i javni *blockchain*. Privatni *blockchain* se koristi u slučajevima kada informacije želimo dijeliti samo sa određenim ljudima, dok se javni koristi kada želimo da informacije zapisane u lancu budu javno dostupne.

Blockchain pruža maksimalnu zaštićenost podataka korištenjem kriptografskih metoda.

Svaka transakcija, jednog ili niza zapisa, pretvara se u *hash* vrijednost, koju je vrlo lako izračunati ako nam je poznata ulazna vrijednost, te na taj način provjera vrijednosti je uvijek prilično jednostavna. Da bih zapisi bili konzistentni zaslužni su algoritmi za postizanje konsenzusa od kojih su najpoznatiji *proof of work, proof of stake* , te *proof of elapsed* time. U slučaju napada, napadač mora kompromitirati minimalno 51% lanca kako bih napad uspio, što je ostvarivo na primjerima manjih lanaca, no u slučajevima većih lanaca to je teško ili nemoguće ostvariti s obzirom da je potrebna iznimno velika procesorska moć.

Blockchain tehnologija nam omogućuje ravnopravnost svih entiteta u lancu, te sigurnost i povjerenje.

2. *Blockchain* tehnologija

2.1. Što je *blockchain*

Blockchain tehnologija predstavlja distribuiranu listu digitalnih informacija podijeljenu između svih čvorova u lancu. *Blockchain* tehnologija nastala je 2008. godine, a njen je kreator pojedinac ili grupa ljudi predstavljajući se pod pseudonimom Satoshi Nakamoto[1]. Osmišljena je za potrebe distribuiranja digitalnog novca Bitcoin. Prekretnica u svijetu dogodila se krajem 2017. godine kada je cijena jednog Bitocin-a dosegla astronomsku cijenu od 20000$. Tada su svi mediji počeli špekulirati o do tada nepoznatoj valuti. Kreatori mnogih drugi kriptovaluta kasnije iskoristili su *Blockchain* tehnologiju kako bi napravili kriptovalutu koja bi na tržištu mogla konkurirati Bitcoinu. Korištenjem *blockchaina* i kriptografije postigli su sigurni transfer digitalnog novca. *Blockchain* predstavlja glavnu knjigu u kojoj su zapisane sve transakcije. Svaki čvor unutar *blockchaina* ima svoju kopiju knjige koja se ažurira svaki puta kada se nova transakcija zapiše u *blockchain*. *Blockchain* sustav je decentralizirani sustav, u kojem nema središnjeg autoriteta. Sustav je izgrađen po modelu ravnopravnih partnera (engl. *peer to peer*). Svi čvorovi sustava mogu zapisivati nove zapise i čitati postojeće. Za provjeru svakog novog zapisa koji se želi zapisati u *blockchain* koriste se algoritmi za postizanje konsenzusa. To je sustav koji osigurava da se svi čvorovi slažu da je određeno stanje sustava istinsko stanje, osigurava da su podaci u knjizi svakog čvora jednaki, te sprječava zlonamjerno manipuliranje podacima. *Blockhain* rješava problem povjerenja među entitetima u lancu, te osigurava transparentnost i sigurnost krajnjih kupaca.

Danas se ova tehnologija koristi u raznim industrijama kao što su financijski sektor, prehrambena industrija, medicina i dr. zbog spoznaje o sigurnosti, iskoristivosti, te profitabilnosti koju pruža.

Glavne karakteristike *Blockchain* tehnologije su:

- Nepromjenjivost
- *Peer to peer* mreža
- Transparentnost
- Konsenzusni algoritam
- Binarno *hash* stablo (engl.*Merkle tree*)
- Kriptografija

2.2. Dijelovi *blockchaina*

2.2.1. Struktura bloka

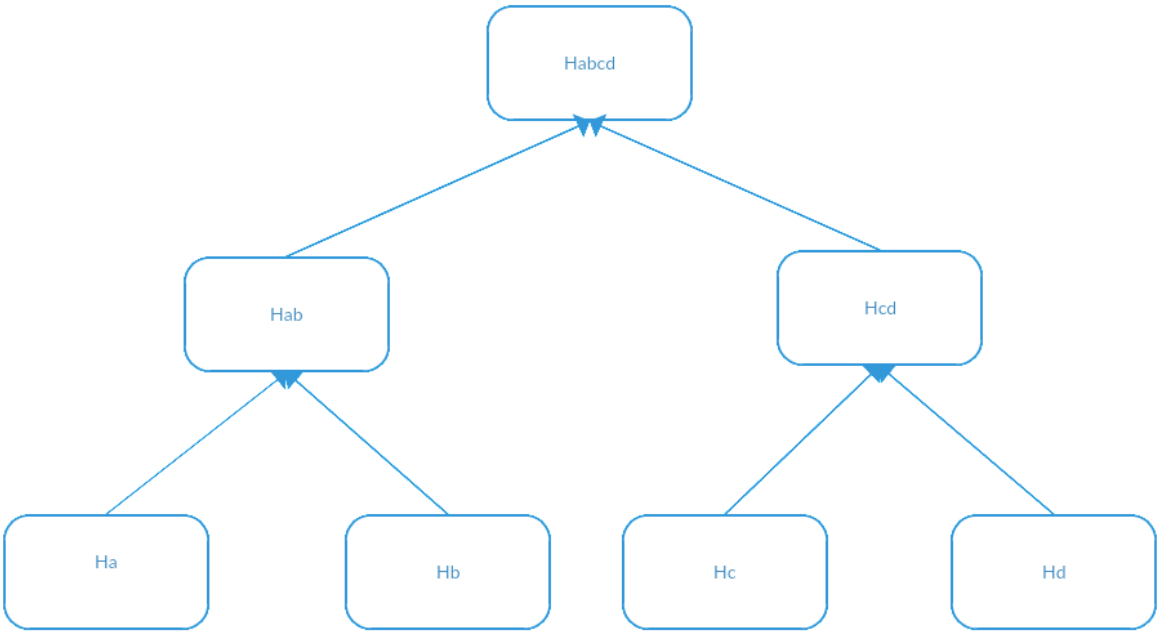
Blockchain se sastoji od blokova. Blok je struktura podataka u kojoj se zapisuju digitalne informacije koje se onda dalje dijele putem *blockchaina.* Blok se sastoji od zaglavlja bloka i liste transakcija koje prenosi. Zaglavlje bloka u sebi sadrži metapodatke. U tim metapodacima nalazi se:

1) Verzija – svaki čvor koji koristi Bitcoin protokol mora implementirati istu verziju koja je zapisana u ovom polju
2) Korijen binarnog *hash* stabla – kriptografski *hash* koji u sebi sadrži informacije o svim zapisima unutar tog bloka
3) *Hash* prethodnog bloka – ovo je digitalni otisak(*hash*) zaglavlja prethodnog bloka
4) Nonce – broj koji određuje algoritam za dodavanje novog bloka u *blockchain*
5) Vrijeme – vrijeme kada je kreiran blok
6) Težinska oznaka – težina algoritma koji rudari moraju riješiti da bi se blok dodao u *blockchain* [3]

Hash prethodnog bloka je digitalni otisak zaglavlja prethodnog bloka u *blockchainu,* a izračunava se tako da se uzimaju sve vrijednosti zapisane u poljima zaglavlja, te se nad njima odrađuje kriptografska funkcija (SHA-256) dva puta preraspodjelom bajtova određenih polja zaglavlja. *Nonce* i težinska oznaka su podaci koji određuju težinu algoritma koji je potrebno riješiti kako bi se novi blok dodao u *blockchain* [18].

2.2.2. Binarno *hash* stablo

Binarno *hash* stablo je skup podataka istog tipa, u kojem svaki čvor može imati djecu. Ako nema djecu, što znači da nema razine ispod njega, takav čvor nazivamo list. Listovi su *hash* vrijednosti nekih podataka. U slučaju kada čvor nije list nego ima djecu, *hash* vrijednost u tom čvoru nastaje spajanje vrijednosti zapisanih u njegovoj djeci, te taj zapis još jednom kriptiran *hash* funkcijom [16]. Kada govorimo o Bitcoin sustavu, tada su u listovima zapisi transakcije, a kao *hash* algoritam tj. funkcija koristi se SHA-256 algoritam. Korijen binarnog *hash* stabla sadržanog u zaglavlju bloka je zapravo pojednostavljeno *hash* stablo sa samo jednim čvorom. Čuva se samo korijen radi skalabilnosti i brzine sustava, jer možemo zamisliti koliko bi bajtova morao sadržavati svaki čvor kada bi u sebi sadržavao cijelo stablo.



Slika 1. Binarno *hash* stablo

2.2.3. Kriptografska *hash* funkcija

Kriptografska *hash* funkcija je funkcija koja za ulaz proizvoljne veličine, daje uvijek izlaz fiksne veličine. Na primjer u slučaju SHA-256 kriptografske funkcije, ulaz može biti proizvoljno velik dok je izlaz uvijek 256 bitova tj. 32 bajta. Svaka promjena na ulaznim podacima, pa čak i samo jednog bita, uzrokovat će drugačiji izlaz, dok za isti ulaz izlaz je uvijek isti. Zbog te činjenice, jako je teško iz izlaza dobiti ulaz, dok iz ulaza dobiti izlaz i provjeriti dali je informacija na izlazu točna je iznimno jednostavno. Kriptografska *hash* funkcija je posebna klasa *hash* funkcija koja ima različita svojstva čineći je tako idealnom za kriptografiju [4]. Postoji niz svojstava koje kriptografska funkcija mora zadovoljiti kako bi bila sigurna, a to su:

a) Deterministično svojstvo
b) Brz izračun
c) Otpor prije slike (engl. *Pre-Image Resistance*)
d) Izmjena jednog bita – drugačiji *hash*
e) Jedinstvenost

a) Deterministično svojstvo

Deterministično svojstvo nam govori da bez obzira koliko puta neki podatak postavimo na ulaz, uvijek ćemo dobiti isti izlaz. To nam je ključno, jer da svaki puta dobijemo drugi izlaz ne bih bili u mogućnosti pratiti ulaz

b) Brz izračun

Ovo svojstvo je iznimno važno, jer u slučaju da je izračun *hashirane* vrijednosti spor, cijeli *blockchain* sustav ne bih bio upotrebljiv.

c) Otpor prije slike (engl. *Pre-Image Resistance*)

Svojstvo koje nam opisuje da je gotovo nemoguće otkriti vrijednost A kada na njega primijenimo *hashiranu* funkciju te dobijem H(A). Moguće je u slučajevima malih zapisa ili poznatih mogućnosti, no ako se radi o velikom zapisu tu nastaje problem. Tada se koristi brutalna sila (engl. *Brute force*) metoda u kojem uzimamo nasumično odabrani ulaz, *hash*iramo ga te ga uspoređujemo sa željenom *hash* vrijednosti koju pogađamo, no ova metoda je neisplativa.

d) Izmjena jednog bita – drugačiji *hash*

Promjenom samo jednog znaka u rečenici, na primjer umjesto točke na kraju rečenice postavimo zarez, *hash* vrijednost više neće biti ista.

Rečenica	SHA-256
Ovo je jedna jednostavna rečenica.	600148B9FA563D708BBE7B1DB763E5E81960BD854F035F8B0949EF5383E03CD4
Ovo je jedna jednostavna rečenica,	BC02E2C51D051EDAC18370DF8E9E693B9F1F14677124E78900B939EF09DEE1F6

Tablica 1. Prikaz kriptiranih rečenica sa inicijalnom razlikom jednog znaka

e) Jedinstvenost

Ako su dva različita ulaza A i B *hash*irani u H(A) i H(B) s velikom vjerojatnošću možemo reći da nije moguće da je H(A) = H(B). Postoji mogućnost, no isprobavanje svih mogućih verzija oba ulaza je neisplativo.

N-bitni *hash* je mapa iz poruke proizvoljne duljine do n-bitnih *hash* vrijednosti. N-bitni kriptografski *hash* je jednosmjeran i otporan na koliziju. Za pronalazak poruke iz *hash* vrijednosti to bi zahtijevalo 2^n *hash* izračuna bilo koje poruke da za izlaz daje traženu *hash* vrijednost. Svojstvo otpornosti na koliziju u brojkama znači da pokušamo pronaći bilo koje dvije poruke koje *hashirane* daju jednaki *hash. Z*a takvu operaciju potrebno je $2^{n/2}$ *hash* izračuna. Funkcioniranje *hash* funkcije u Bitcoinu opisati ću primjerom *hashiranja* rečenice „Ovo je jednostavna rečenica“. Na kraj rečenice dodat ću broj i gdje je i = 0,1,2,3,4… sve dok *hash* ne bude započinjao brojem 0. *Hash* algoritam biti će SHA-256.

Rečenica	SHA-256
Ovo je jedna jednostavna rečenica0	5ee80b90f1524409f50946d4575f12bc356269482a72563ba9c0c28c0726baba
Ovo je jedna jednostavna rečenica1	895cf384934be359bf84380f2b6272b16e4a483e51b162dae4c6999be9e25b13
Ovo je jedna jednostavna rečenica2	a9b52361b60044c6773ae84880790556091c2529fcf698a758f60dfe8f929a9b
Ovo je jedna jednostavna rečenica3	8c2b978ed4088f738fdbaa6fe5cd84bb1e1245bb90e91e3149e61c4e0eeee28d
Ovo je jedna jednostavna rečenica4	c226bf6dcf80e6a3878230b0619be9441f04a202cd4134f61e42f979a0dd28a4
Ovo je jedna jednostavna rečenica5	f098d0254921c199364292550ee65f121c6bc7cb39cb269974ce95b768621834
Ovo je jedna jednostavna rečenica6	0bcf1d7b7dbd9b1b3ff10097ed0bcb4674e76fc470da25787e384ed1fdbbcb28

Tablica 2. Prikaz ručnog rudarenja

Rudari obavljaju identičnu radnju kao mi u gornjem primjeru. No, broj *nonce* (broj znamenki 0 u slučaju Bitcoin-a) znamenki na početku *hash*-a ovisi o težinskom broju. Nama je u slučaju sa jednom znamenom 0 trebalo 6 pokušaja, dok bi nam za veći broj znamenki 0 na početku trebalo puno više pokušaja. Rudar koji prvi pronađe rješenje objavljuje svima u mreži da je došao do rješenja, kako bih ostali partneri rudari prestali sa pogađanjem iste.

2.2.4. SHA-256

SHA-256 (engl. *Secure Hash Alghoritam*) je jedna od kriptografska *hash* funkcija osmišljena od strane National Security Agency (NSA) u sklopu SHA-2 obitelji [12]. 256 označava duljinu *hash* vrijednosti dobivene iz proizvoljne duljine poruke koja se *hashira*.

U nastavku prikazati ću računanje *hash* vrijednosti iz poruke „SAT“.

Da bismo pripremili poruku da bude *hashirana* potrebna su nam dva koraka [11]:

1.) Proširiti poruku da bude djeljiva sa 512
2.) Razdijeliti poruku u 512-bitne blokove $M^{(1)}$, $M^{(2)}$,…, $M^{(N)}$

1.) Proširivanje poruke

1.) Duljina poruke M u bitovima označujemo s „d“
2.) Na kraj poruke dodajemo bit „1“
3.) Na kraj poruke dodajemo k puta broj 0, a k je najmanji pozitivni broj jednadžbe d+1+k =448 mod 512
4.) Na kraj poruke dodajemo 64-bitni blok koji je prikaz broja d u binarnom obliku

01110011	01100001	01110100	1	000…0	000…0011000
S	A	T	1	k puta 0 (423)	d binarno (64)

Tablica 3. Prikaz proširene poruke M

2.) Razdjeljivanje poruke na blokove

Nakon proširenja poruku M potrebno je podijeliti na N 512-bitnih blokova $M^{(1)}$, $M^{(2)}$,…, $M^{(N)}$. Prvih 32 bita spremamo u $M_0^{(i)}$, sljedećih 32 bita u $M_1^{(i)}$, i tako do $M_{15}^{(i)}$. Blokovi poruke obrađeni su jedan po jedan, po formuli:

$$H^{(i)} = H^{(i-1)} + C_{M^{(i)}}\left(H^{(i-1)}\right),$$

gdje je C SHA-256 kompresijska funkcija. H^N je *hash* od poruke M.

Oznaka	Naziv
$\wedge$	Operator I
$\urcorner$	Operator komplement
$\oplus$	Operator XILI

+	Modulo 2^{32} zbrajanje
R^n	Pomicanje u desno za n bitova
S^n	Rotiranje u desno za n bitova

Tablica 4. Operatori koji se koriste za izračun SHA-256 algoritma

Šest logičkih funkcija koriste se u SHA-256. Svaka od tih funkcija uzima 32-bitnu riječ i izračunava 32-bitnu riječ kao izlaz. Ovo su tih 6 funkcija:

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge y)$$

$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

$$\sum_0(x) = S^2(x) \oplus S^{13}(x) \oplus S^{22}(x)$$

$$\sum_1(x) = S^6(x) \oplus S^{11}(x) \oplus S^{25}(x)$$

$$\sigma_0(x) = S^7(x) \oplus S^{18}(x) \oplus R^3(x)$$

$$\sigma_1(x) = S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x)$$

Kod algoritma SHA-256 izgleda na sljedeći način:

Za i = 1,…,N (N broj blokova veličine 512 bitova proširene poruke M)

{

- Inicijaliziraj registre a, b, c, d, e, f, g, h sa $(i-1)^{st}$ srednjom *hash* vrijednosti(uzima se inicijalna *hash* vrijednost ako je i = 1)
- $a \leftarrow H_1^{(i-1)}$
  $b \leftarrow H_2^{(i-1)}$
  …
  $h \leftarrow H_8^{(i-1)}$

- Primjeni SHA-256 kompresijsku funkciju da bi ažurirao registre a, b, c …,h

Za j = 0,…, 63

{

Izračunaj $C_h(e, f, g)$, Maj(a, b, c), $\sum_0(a)$, $\sum_1(e)$ i Wj (pogledaj izračun niže)

$T_1 \leftarrow h + \sum_1(e) + C_h(e, f, g), + K + W_j$

$T_2 \leftarrow \sum_0(a) + Maj(a, b, c)$

$h \leftarrow g$

$g \leftarrow f$

$f \leftarrow e$

$e \leftarrow d + T_1$

$d \leftarrow c$

$c \leftarrow b$

$b \leftarrow a$

$a \leftarrow T_1 + T_2$

}

- Izračunaj i^{th} srednju *hash* vrijednost $H^{(i)}$
- $H_1^{(i)} \leftarrow a + H_1^{(i-1)}$
  $H_2^{(i)} \leftarrow b + H_2^{(i-1)}$
  …
  $H_8^{(i)} \leftarrow h + H_8^{(i-1)}$

}

$H^{(N)} = (H_1^{(N)}, H_1^{(N)},\ldots, H_8^{(N)})$ je *hash* od poruke M.

$W_j = M_j^{(i)}$ za j = 0,1,…,15, i za

j = 16 do 64

{

$W_j \leftarrow \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16}$

2.3. Decentralizirani sustav

Decentralizirani sustav omogućuje ravnopravnost svih čvorova u mreži. Kod decentraliziranog sustava ne postoji središnji autoritet koji donosi odluke, već se odluke donose na temelju odluka svih čvorova uključenih u proces [5]. Takav sustav je pravedniji i

sigurniji od centraliziranog sustava. Sustav djeluje na način da ako nekom od sudionika treba pomoć, on pomoć potražuje od svog susjeda, te ako ni oni ne uspijevaju riješiti problem, povezuju se dalje sa susjedima sve dok ne skupe dovoljno čvorova i kompjuterske snage za prijenos informacije. Jedna od najvažnijih prednosti ovakvog sustava je da nema jedinstvenog mjesta pada, jer niti jedan od sudionika nema najveće ovlasti u procesu. Neke od prednosti još su skalabilnost i neovisnost, u slučaju da neki od čvorova nisu dostupni, umjesto njih se vrlo lako uključuju novi, kako bih se postigla veća kompjuterska snaga [2]. Veća privatnost osigurana je ovakvim sustavom, jer nema centralnog mjesta gdje se svi podaci spremaju.

Centralizirani sustav izgrađen je oko jednog poslužitelja koji upravlja svim procesima. Prednost ovakvog sustava je konzistentnost i efikasnost. U ovakvom sustavu manje je održavanja, te veća kontrola nad podacima koji mrežom putuju. U slučaju pada ovog poslužitelja, sustav koji se odvija na njemu, može biti privremeno ili trajno nedostupan. Zaključiti ću da se u početnim fazama treba dobro informirati što nam treba, te donijeti dugoročnu odluku u skladu sa našim potrebama.

Sustavi koji koriste *blockchain* tehnologiju koriste decentralizirani sustav ravnopravnih partnera, u kojem se među sudionicima razmjenjuju podaci, te se na taj način isključila potreba za centralnim poslužiteljem.

2.3.1. Novčanik

Novčanici u *blockchain* sustavu su zapravo programska rješenja koja omogućuju korisnicima kupovinu, prodaju i praćenje stanja svojeg digitalnog novca ili nekih drugih vrijednosti [6]. U novčaniku su zapisane sve transakcije. Svaka transakcija u *blockchainu* je bazirana na asimetričnoj kriptografiji koja koristi dvije vrste ključeva:

1) Javni ključ
2) Privatni ključ

Vrijednosti ovih ključeva nisu jednake. Javni ključ možemo davati bilo kome, dok privatni ključ trebamo držati na sigurnom mjestu. Privatni ključ možemo poistovjetiti sa jedinstvenim četveroznamenkastim pinom naše bankovne kartice, a javni ključ kao broj našeg računa. Kada nam netko želi poslati digitalni novac na naš račun, on će potpisati tu transakciju našim javnim ključem, dok će novac sjesti baš na naš račun jer je privatni ključ

zapisan unutar novčanika, te na taj način jedino mi možemo sa svojim privatnim ključem potvrditi javni ključ. Kada se dogodi da se potvrde javni i privatni ključ, transakcija se zapisuje u *blockchain*, a novac se dodjeljuje nama.

Binarno *hash* stablo omogućuje novčanicima provjeru kojem bloku u *blockchainu* pripada određena transakcija. S obzirom da novčanik poznaje sadržaj transakcije vrlo lako može saznati i *hashiranu* vrijednost te transakcije, koju onda predaje binarnom *hash* stablu.

Vrste novčanika:

1) Desktop novčanik
2) Novčanik za mobilne uređaje
3) *Hardware* novčanik
4) Novčanik u oblaku

1) Desktop novčanik je novčanik instaliran na naše računalo ili laptop. Ova vrsta novčanika je sigurnija od novčanika u oblaku. Potrebno je imati dobar antivirusni program, te je potrebno ponekad povezati se na internet kako bi se ažurirali podaci o transakcijama u *blockchainu*.

2) Novčanik za mobilne uređaje je lagana verzija novčanika pogodna za instaliranje na mobilni uređaj ili tablet. Također važno je na uređaju imati dobru zaštitu kako bi se zaštitili od krađe digitalnog novca sa našeg novčanika.

3) Hardware novčanik je najsigurniji novčanik. Novčanik nije potrebno povezivati na Internet, a bez obzira na to on i dalje može potpisivati transakcije.

4) Novčanik u oblaku pruža najjednostavniju, ali i najrizičniju mogućnost korištenja novčanika. Za kreiranje je dovoljna email adresa i lozinka. Činjenica da je novčanik u oblaku i da je internetska veza konstantno u upotrebi dovela je do velikog broja lažnih internetskih stranica sa jako sličnim sučeljem, gdje naivni korisnik unosi svoju email adresu i lozinku misleći da je na pravoj stranici, te nakon toga ostaje bez svega što je posjedovao u svom novčaniku.

Postoje razni načini kako se osigurati od prijevare i gubitka digitalno novca. Neki od novčanika nude mogućnost odabira 12 riječi koje će kasnije služiti kao mogućnost dohvaćanja zaboravljene lozinke, postavljanje podsjetnika za lozinku (engl. *Password hint*), prevencija od neautoriziranog pristupa novčaniku kreiranjem dva-faktora autorizacije (engl. *Two-factor authentication*) koja omogućuje spremanje broja mobilnog uređaja, na koji onda

dobivamo jednokratnu lozinku za pristup. Najbitnije je sačuvati privatni ključ jer u slučaju gubitka ili krađe istoga, ostajemo bez svega.

2.3.2. Rudari

Rudari su računala koja preuzimaju transakcije od novčanika i pohranjuju ih na *blockchain*. Rješavajući neki algoritam pomoću specijalnog programa, u Bitcoin slučaju taj se algoritam naziva *proof of work*. Svojom kompjuterskom snagom omogućuju zapisivanje novih transakcija na *blockchain*, te za uzvrat dobivaju određeni iznos Bitcoina. Da bi se izrudario novi Bitcoin potrebno je vrijeme i kompjuterska snaga. U početcima su rudari koristi moć procesora, no kasnije su shvatili da su grafičke kartice koje se inače koriste za igranje igrica, puno efikasnije, ali isto tako troše puno električne energije. Nerijetko se pojedini rudari priključuju u rudarski bazen (engl. *Minning pool*) kako bih što brže, uz pomoć drugih, rješavali algoritme, te tako lakše dolazili do digitalnog novca, dijeleći ga ravnopravno sa svima u bazenu. Nakon što potvrde ispravnost transakcije, spajaju transakcije u blok, nakon toga uzimaju zaglavlje zadnjeg bloka, dodaju ga u novi blok kao *hash* vrijednost. Kako bi dodali novi blok moraju riješiti *proof of work* algoritam, te nakon toga postavljaju blok u *blockchain.* Težina rudarenja ovisi o vrijednosti težine rudarenja (engl. *Minning difficulty*), a odnosi se na težinu algoritma koji je potrebno riješiti kako bi se dodalo novi blok u *blockchain* u ovom trenutku, u odnosu na prvi ikada izrudareni blok.

Proces rudarenja novih blokova u sustavu je kružni a sastoji se od sljedećih pravila:

- Sve više rudara se spaja u mrežu
- Povećava se broj potvrđenih blokova
- Vrijeme potrebno za potvrdu bloka se smanjuje
- Sustav automatski povećava težinu rudarenja
- Smanjuje se broj potvrđenih blokova

Što više rudara se spaja na mrežu, povećava se broj potvrđenih blokova. U tim trenutcima vrijeme potrebno za rudarenje bloka se smanjuje (u Bitcoin sustavu idealno vrijeme rudarenja je 10 minuta)

. Sustav automatski povećava težinu rudarenja. Tada se smanjuje broj potvrđenih blokova, a vrijeme potrebno za potvrđivanje bloka vraća se u normalu, i tako u krug.

3. Algoritmi u *blockchain* tehnologiji

Algoritmi u *blockchain* tehnologiji ili algoritmi za postizanje konsenzusa su procesi u računarstvu koji se koriste za postizanje dogovora o jednoj vrijednosti podataka među distribuiranim procesima ili sustavima. Konsenzus je dinamičan način postizanja dogovora one cjeline koja je najbolja za ostale u sustavu. Algoritmi za postizanje konsenzusa osmišljeni su kako bi se podigla pouzdanost u mreži koja uključuje više nepouzdanih čvorova.

Ovakvi algoritmi pod pretpostavkom da će neki procesi i sustavi biti nedostupni, te da će neke transakcije biti izgubljene, potražuju odgovor od minimalno 51% čvorova u mreži kako bi transakcija bila ispravna.

Ciljevi algoritama za postizanje konsenzusa su sljedeći:

a) Dogovor – mehanizam prikuplja sve sporazume iz grupe koliko god to može
b) Kolaboracija – Svaka skupina nastoji postići bolji dogovor koji rezultira interesima skupine u cjelini
c) Suradnja – Svaki pojedinac će dati sve od sebe i staviti svoje interese sa strane
d) Jednaka prava – Svaki sudionik ima istu vrijednost prilikom glasovanja.
e) Sudjelovanje – Svatko unutar mreže mora sudjelovati u glasovanju
f) Aktivnost – Svaki član grupe je jednako aktivan, ne postoji netko sa više ili manje odgovornosti

3.1. Problem usuglašavanja bizantskih generala

Zamislimo sljedeću situaciju. Vojska zločinaca zarobila je dvorac sa ukupno 300 ratnika. Vojska oslobođenja ima 500 ratnika, te su podijeljeni u 5 grupa po 100 ratnika od kojih je jedan u svakoj grupi general. 500 ratnika protiv 300 ratnika, rezultat bi trebao biti na strani osloboditelja, no u slučaju da ne napadnu svi u isto vrijeme, postoji mogućnost da izgube bitku. U to vrijeme Bizanta nije postojao niti telefon, niti mobitel ili bilo što što bi se moglo danas koristiti za komunikaciju. Poruka u kojoj piše „Napadamo u 9h“ biti će prenesena preko jahaća koji će svakom od pet generala poruku osobno uručiti. Svaki od generala može biti zao čovjek, te lažirati poruku kako ne bih svi napali u isto vrijeme. Zamislimo da je

general broj 3 izdajnik. Prvi general napisao je „Napadamo u 9h“, drugi je također napisao istu poruku, no treći general je napisao „Napadamo u 8h“. U ovom scenariju general 3,4 i 5 napadaju u 8h te će vjerojatno izgubiti bitku, te kada napadnu preostala dva generala sa ukupno 200 vojnika također će s velikom vjerojatnošću izgubiti. U scenariju u kojem želimo prikazati sigurnost *blockchain* tehnologije, isti ovaj scenarij bi izgledao na sljedeći način.

Svaki general koji želi poslati poruku ima dva pravila:

1) Da bih poruka bila vjerodostojna, mora proći 10 minuta
2) Svaka poruka mora sadržavati povijest svih prethodnih poruka

Čak i u slučaju da je general 1 izdajnik, on piše prvi poruku, što god da napiše biti će istina. Kada general 2 piše poruku trebati će mu 10 minuta i mora uključiti i poruku generala 1 na list papira. Dolazim do generala 3 koji je izdajnik. Da bi uspio lažirati vrijeme svih prethodnih poruka treba mu 20 minuta za njihove poruke i 10 minuta za svoju, što je 30 minuta, a naveli smo da je vrijeme dozvoljeno za vjerodostojnu poruku 10 minuta. Jedino što mu preostaje je napisati istinitu poruku i ne otići u to vrijeme u obranu dvorca, što bi za grupu značilo 100 vojnika manje. Sa brojčanom prednošću od 200 vojnika više, osloboditelji će sigurno pobijediti.

U Bitcoin svijetu svakih 10 minuta se odobrava jedna transakcija koje je ispravna da se upiše u *blockchain*. Desetominutni rad koji smo ovdje spomenuli odnosi se na *proof of work* algoritam. To je težak posao koji (iziskuje uloženo vrijeme), ali je vrlo lagano provjeriti jesu li zapisi ispravni. Primjer jednog takvog rada možemo zamisliti kao zapisivanje brojeva na komad papira od 1 do 1000. Osoba koja obavlja taj zadatak mora uložiti puno više vremena za napisati te brojeve, nego mi koji ćemo kasnije kontrolirati ispravnost poruke [8].

3.2. Lamportov algoritam

Lamportov algoritam jedan je od najpoznatijih algoritama za postizanje konsenzusa u sinkroniziranoj mreži. Objasniti ćemo ovaj algoritam na primjeru usuglašavanja bizantskih generala. Generali se dogovaraju oko odluke napasti ili povući se, što znači da donose odluke veličine jednog bita, 1 ako napadaju, 0 ako se povlače. Dok postavljamo algoritam koji će riješiti njihov problem, najbitnije je da ga oblikujemo na način da svi koji odabiru istu akciju imaju isti ulaz. Lamportov algoritam zahtjeva da je $N > 4f$. N je broj sudionika, dok je f broj

izdajnika u sustavu. Svaki od partnera svoju odluku definira sa 0 ili 1, koja se može promijeniti. Na kraju svi usuglašeni generali moraju imati istu poruku. Algoritam je zasnovan na rotirajućem koordinatoru kralju. Svaki partner donosi zasebnu odluku, ali u lokalno polje A[ ] zapisuje svoj prijedlog i prijedlog svih drugih. Kralj donosi kraljevski prijedlog na osnovu većine prijedloga u polju ili ako nema većine onda postavlja inicijalnu vrijednost kao svoj prijedlog. Svi partneri čuvaju kraljevu odluku u varijabli kraljevaOdluka.

Svaki partner donosi svoju odluku i sprema je u varijablu mojaOdluka. Partner ima opciju izabrati između prijedloga koji dominiraju u polju ili izabrati kraljevu odluku. Partner će odabrati prijedlog većine ako se ona pojavljuje u polju na N/2 + f mjesta. U našem slučaju, gdje nam je poznato da je samo jedan general izdajnik, dogovor između generala je moguće postići. Rješenje jednadžbe, gdje je N = 5 (ukupno 5 generala), f = 1 (jedan izdajnik), iznosi 5 > 4*1.

3.3. Proof of work

Proof of work konsenzusni algoritam nastao je 2004. godine. Osmislio ga je Hal Finney. Prva njegova implementacija bila je 2008. godine za potrebe kriptovalute Bitcoin, te se danas koristi i u drugim kriptovalutama. Proces je poznat kao rudarstvo, a obavljaju ga čvorovi rudari. *Proof of work* dolazi kao odgovor na matematički problem, onaj koji zahtijeva dugotrajan posao da bi se došlo do rezultata, ali je rezultat vrlo lako provjeriti. Rudari pokušavaju pogoditi *hash* vrijednost transakcije kako bih mogli provjeriti ispravnost iste, koristeći kompjutersku snagu. Rudari se međusobno natječu tko će prvi potvrditi *hash* i biti nagrađen za svoj rad. Kada rudar uspije riješiti matematički problem, formira se novi blok i u njega se zapisuju transakcije. U slučaju Bitcoina naziv *proof of work* slagalice zove se *Hashcash*. Ovaj algoritam omogućuje promjenu složenosti algoritma na temelju ukupne snage mreže. Prosječno vrijeme rješavanja slagalice je 10 minuta. Glavne prednosti *proof of work* algoritma su otpornost na DDoS (engl. *Distributed Denial-of-Service*) napade. Koraci *proof of work* algoritma su sljedeći:

1) Transakcije se grupiraju zajedno u blok
2) Rudari potvrđuju transakcije unutar blokova kao legitimne
3) Rudari zatim rješavaju matematički zadatak
4) Rudari dobivaju nagradu za odrađeni posao
5) Potvrđene transakcije dodaju se u javni *blockchain*

Da bismo mogli rudariti potrebna nam je velika jačina računalnih resursa. Kada *proof of work* ne bi postojao, dodavanje novih blokova sveli bismo na najbrže računalo u mreži, te bi tada izgubili decentraliziranost sustava. U slučaju Bitcoina rudarenje bloka je teško jer SHA-256 *hash* težinske vrijednosti zaglavlja i broja *nonce* mora biti manji ili jednak ciljnom *hashu* kako bi blok bio prihvaćen od strane mreže. *Nonce* se povećava kako bi vrijeme potrebno za generiranje novog bloka bilo oko 10 minuta.

4. Područja i primjeri upotrebe *blockchain* tehnologije

Područja primjene *blockchaina* su razne. U bilo kojem području gdje je potrebno riješiti problem povjerenja između entiteta, *blockchain* se nameće kao idealno rješenje. Kao što sam već naveo, jednom zapisani podaci zauvijek ostaju zapisani bez mogućnosti izmjene. Samo neki od primjera upotrebe *blockchaina* su :

1) Prehrambena industrija
2) Medicina
3) Bankovni sustav
4) Glasački sustav

Primjena *blockchaina* u prehrambenoj industriji već se koristi u svim u sustavu Walmart dućana u Americi. Praćenje svakog proizvoda od njegove izrade, preko prijevoza, do svakog dućana omogućuje svim kupcima njihovih dućana sigurnost i provjerenu kvalitetu svake namirnice. Kroz svoj praktični dio rada pokazati ću na primjeru jedne namirnice kako se može iskoristiti *blockchain* za ovakvu potrebu.

U medicini se *blockchain* primjenjuje za zapisivanje svih ikada obavljenih pregleda, recepata te povijesti bolesti pacijenta. Takvi zapisi u *blockchainu* postaju nepromjenjivi, te dostupni svakom doktoru u realnom vremenu, bez potrebe za prenašanje velike količine papira od doktora do doktora ili od bolnice do bolnice.

Bankovni sustav kakav danas poznajemo koristi centralizirani sustav. Sistem koji bi se tamo primijenio koristio bi privatni *blockchain*, u kojem bih mogli sudjelovati samo autorizirani članovi. Prijenos novca bio bi sigurniji i brži, te ne bih trebali koristiti centralni sustav koji kontrolira sredstva.

Glasački sustav izgrađen na *blockchain* tehnologiji bio bi prava revolucija današnjem sustavu. S obzirom da jedna osoba može samo jednom glasati, njen glas bi bio trajno zabilježen i nikada nepromijenjen. Ljudi bi mogli glasati iz vlastitog doma. Kada bi takav sustav postojao, više ne bi postala mogućnost namještanja glasova, te bi brojanje glasova bilo brže i efikasnije.

4.1. Kriptovalute

U nastavku opisati ću nekoliko zanimljivih primjena *blockchain* tehnologije za stvaranje kriptovaluta. Kriptovalute (engl. *Cryptocurrency*) su digitalni novac koji se prenosi između čvorova u mreži. Najvažnije karakteristika kriptovaluta je ne postojanje središnjeg autoriteta, nego se koristi *blockchain* kako bi se omogućio potpuno decentralizirani sustav [7]. Korisnici koji žele kupiti određene kriptovalute dolaze do njih preko mjenjačnica na internetu, te tako ulažu stvarni novac u zamjenu za njih. Novi novčići (engl. *Coins*) nastaju rudarenjem.

Ljudi mogu doći do svojih kriptovaluta kupovanjem istih preko raznih online mjenjačnica, rudarenjem, prodajom usluga ili proizvoda u zamjenu za kriptovalutu. Uvelike se spominje korištenje kriptovaluta za potrebe crnog interneta (engl. *Dark web*) iz razloga što su osobe koje trguju na crnom internetu više zaštićene od izlaganja osobnih informacija na internetu trgujući kriptovaluta, nego pravim novcem i pravim podacima svog računa. Bitcoin je prva kriptovaluta na svijetu kreirana 2008.godine. Nakon generiranja novog bloka, rudar za njega dobiva nagradu, a novi novčići su kreirani.

Potrebno vrijeme je 10 minuta, ukupna količina Bitcoin novca je 21 milion Bitcoin-a. Bitcoin algoritam je objavljen na internetu kao OpenSource kod (svatko može analizirati tijek pojedine metode, pisati svoje metode, te prilagođavati kod svojim potrebama), te su na temelju toga nastale mnoge druge kriptovalute koje danas poznajemo.

Ripple

Ripple je nastao 2012. godine. Ripple je platforma za prijenos novca i kriptovaluta u jednom. Izvorni kod je objavljen na Internetu (engl. *Open source*). Ripple kao i Bitcoin koristi *proof of work* algoritam. Temelji se na zajedničkoj javnoj bazi podataka koja koristi konsenzusni algoritam između poslužitelja za validaciju kako bi se očuvao integritet. Takvi poslužitelji mogu pripadati bilo kome ili primjerice banci.

Ripple protokol (token predstavljen kao XRP) namijenjen je prijenosu novca između dvije strane. Ripple ima za cilj omogućiti sigurne, besplatne financijske transakcije bilo koje veličine bez naplate [9].

IOTA

IOTA je projekt napravljen na javnom *blockchainu*, koji kao rješenje nudi kriptovalutu koja bi se koristila između IOT uređaja. Prva je *OpenSource* distribuirana glavna knjiga koja želi izgraditi svijet IOT uređaja koja bi omogućavala mikrotransakcije i podatkovni integritet uređaja. Korištenjem *blockchaina* omogućuju sustav bez treće strane, te tako izbjegavaju plaćanje naknada i sporosti sustava. Glavna knjiga zove se Tangle, koju nazivaju poveznicom između Web 3.0. i Internet svega (engl. *Internet of Everything*). Žele postići *Machine Economy* odnosno komunikaciju IOT uređaja bez da o tome treba brinuti čovjek. Kao primjer naveo bih plaćanje parkinga, kada bi se osoba parkirala, automobili bi plaćali parking automatu za naplatu parkinga, bez da čovjek mora o tome razmišljati ili plaćati ga osobno.

U IOTA sustavu nikakve dodatne naknade se ne plaćaju prilikom transakcije. Rudari moraju potvrditi prethodne dvije transakcije, te svi rudari dobivaju identičnu nagradu za svoj rad.

Litecoin

Kreator ove kriptovalute je Charles Lee, a stvorio ju je 2011. godine. Litecoin kod je OpenSource. Litecoin novčića postoji 84 miliona. Vrijeme potrebno za rudarenje jednog novčića iznosi 2,5 minuta. Kada rudar kreira novi blok za uzvrat dobiva 50 Litecoin novčića. Koristi konsenzusni algoritam proof of work, no u usporedbi sa Bitcoinom ne koristi SHA-256, nego algoritam po nazivu Scrypt. Scrypt za razliku od SHA-256 ne obavlja radnje paralelno. Scrypt se još naziva i teški problem za memoriju. Zbog zagušenja mreže i sporog rudarenja blokova, srednja vrijednost potvrde transakcije može biti i do 30 minuta.

Cardano

Cardano je izgrađen na znanstvenoj filozofiji i recenziranim akademskim istraživanjima.

Osmislio ga je Charles Hoskinson 2015.godine. Postoje tri organizacije koje puno radno vrijeme brinu i razvijaju Cardano, a to su:

- Zaklada Cardano
- IOHK
- Emurgo

IOHK(Input Output Hong Kong) je tvrtka za istraživanje i razvoj koja se zalaže za *peer-to-peer* inovacije *blockchaina* za izgradnju dostupnih financijskih usluga za sve.

Emurgo je japanska tvrtka koja razvija, podržava i provodi komercijalne poduhvate koji žele revolucionirati industriju koristeći tehnologiju *blockchain.* Ove tri tvrtke zajedničkim snagama osiguravaju razvoj Cardano platforme. Cardano tim opisuje Cardano kao treću generaciju *blockchaina.* Prva generacija je Bitcoin i prijenos novca, druga generacija je Ethereum i pametni ugovori, a treća generacija je Cardano. Uzeli su sve što bilo pozitivno napravljeno iz prethodne dvije generacije i dodali neke svoje elemente. Tri glavne karakteristike Cardano-a su:

- Skalabilnost
- Interoperabilnost
- Održivost

Vodili su se filozofskim pristupom te su se usredotočili na skup načela dizajna, najboljih inženjerskih praksi i najboljim načinima istraživanja.

Monero

Monero je siguran, privatni i neprovjerljivi valutni sustav. Koristi posebnu vrstu kriptografije kako bi osigurao da sve njegove transakcije ostaju 100% ne povezive, te da se ne mogu otkriti. Stvoren je 2012.godiine, ali kao Bytecoin. To je bila prva implementacija CryptoNote-a. CryptoNote je protokol aplikacijskog sloja koji pokreće različite decentralizirane valute. Kasnije se Bytecoin ugasio zbog sumnjivih stvari koje su se događale, jedna od njih bila je da se u par mjeseci izrudarilo 80% novčića, te se formirao novi lanac nazvan Bitmonero, kasnije Monero (na Esperanto jeziku znali novčić). Osmislila ga je grupa od 7 programera. Neke od karakteristika Monera su:

- Vaša valuta je samo vaša
- Zamjenjivost
- Dinamična skalabilnost
- ASIC otporan
- Višestruki ključevi

U Monero *blockchainu* imate apsolutnu kontrolu nad vašim transakcijama, te ste vi odgovorni za svoj novac. Budući da je Monero građen na privatnom *blockchainu,* vaš identitet je također privatan, te nitko ne može vidjeti na što trošite svoj novac.

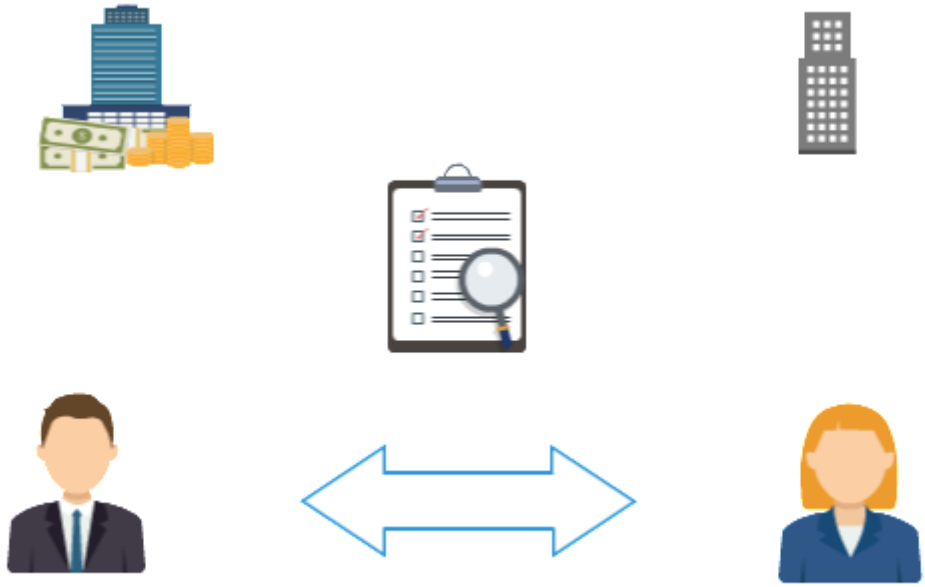
Zamjenjivost u svojoj definiciji znači zamjenjivost dobra ili imovine s drugim pojedinačnim proizvodima ili sredstvima iste vrste, što u prijevodu znači da su svi njegovi podaci i transakcije privatni, nitko ne može znati kakve je transakcije prošao naš Monero prije niti

mogu znati za što je korišten naš Monero prilikom kupnje. Budući da povijest transakcija nikada ne može biti poznata, to znači da trag transakcije ne postoji. Kao rezultat toga koncept „zaraženog“ i „čistog“ Monera ne postoji, stoga se kaže da Monero ima zamjenjivo svojstvo. Kao obrnuti primjer navesti ću Bitcoin, koji nema svojstvo zamjenjivosti, jer se svaki trag zapisuje u pojedinoj transakciji, a s obzirom da je građen na javnom *blockchainu,* sve transakcije su javne te lako provjerljive. Dinamična skalabilnost ovisi o veličini bloka koji se pokušava dodati u *blockchain*. Uzima se prosječna vrijednost zadnjih 100 blokova pod oznakom M100. Ako je veličina trenutnog bloka NBS> M100, tada se nagrada za rudarenje istoga kvadratno smanjuje koliko NBS premašuje M100. Monero ima javni ključ za prikaz i privatni ključ za prikaz, ključ javnog za prikaz koristi se za generiranje jednokratne javne adrese gdje će sredstva biti poslana primatelju, dok privatni ključ za prikaz koristi primatelj za skeniranje *blockchaina* kako bih pronašao sredstva koja mu pripadaju. Postoje privatni i javni potrošački ključevi. Dok su ključevi iz prikaz isključivo vezani za primatelja, potrošački ključevi su vezani isključivo za pošiljatelja. Javni potrošački ključ će pomoći pošiljatelju da pronađe primatelja, te da provjeri potpis ključa. Privatni potrošački ključ služi za kreiranje ključa koji će odobriti pošiljatelju slanje transakcije. Jedna zanimljivost vezana za Monero je da je trenutačno najpoznatija računalna igra na svijetu Fortnite uvrstila ovu kriptovalutu kao način plaćanja dodatnih stvari u računalnoj igri.

4.2. Pametni ugovori

Pametni ugovori ili kako ih još nazivamo samo izvršavajući ugovori, ugovori su koji koriste kriptografiju, digitalni potpis i sigurnost *blockchaina*. Osmislio ih je Nick Szabo 1994. godine. To su ugovori koji se kreiraju i izvršavaju na računalu [17]. Pomoću pametnih ugovora možemo razmjenjivati novac, objekte ili bilo što, što ima smisla pohraniti u ugovoru. U slučaju kada želimo kupiti automobil, naš ugovor sklopljen sa drugom stranom mora potvrditi javni bilježnik, što nam oduzima vrijeme i novac. U slučaju pametnih ugovora, ne postoji treća strana, ugovor je sklopljen između sudionika, te počinje vrijediti od trenutka definiranog u ugovoru, zaobilazeći čekanje, papirologiju ili bilo koji dugotrajan proces koji se danas mora čekati da dobijemo ono što smo platili. Kao primjer naveo bih iznajmljivanje apartmana. Najmodavac i najmoprimac definiraju zajedno stavke ugovora kao što su vrijeme ulaska i izlaska iz apartmana, te iznos najma. Sklopljeni ugovor pohranjuje se na *blockchain,* te se stavke iz ugovora aktiviraju u navedeno vrijeme dolaska

gosta. U trenutku kada dođe vrijeme za ulazak gosta u apartman, gost dobiva lozinku za ulaz u apartman, a najmodavac novac na svoj račun. Ako lozinka ne dospije do gosta u dogovoreno vrijeme *blockchain* osigurava povrat novca. Pametnim ugovorima štedi se novac i vrijeme, te ne postoji mogućnost ljudske pogreške osim u trenutku pisanja ugovora, sigurnost da ćemo dobiti ono što plaćamo osigurana je kopijama na svim čvorovima u mreži kopijom glavne knjige. Postoje i neki nedostaci ovakvog sustava, kao što su pogreške u kodu, država još nije definirala način oporezivanja pametnih ugovora, jednom odrađene transakcije više nije moguće opovrgnuti.



Slika 2. Pametni ugovori

Ethereum

Ethereum je decentralizirana platforma na kojoj se mogu izvršavati pametni ugovori. Pametni ugovori su ugovori sklopljeni između dvije ili više strana, napisani u programskom jeziku Solidity te pohranjeni u *blockchain*, te tako taj ugovor postaje nepromjenjiv, a izvršava se u određenim uvjetima koje smo postavili u ugovoru. Takve aplikacije se zovu decentralizirane aplikacije (engl. *Decentralized applications*) skraćeno dApp. Solidity je programski jezik razvijen od strane Ethereum Community-a sličan Javascriptu. Potrebno je također otvoriti njihov Ethereum novčanik kako bih mogli trgovati sa njihovom kriptovalutom Ethereum. Ethereum trenutno koristi *proof of work* algoritam, no u ovom slučaju ne radi se SHA-256 enkripciji nego o EtHash algoritmu [10]. U skorije vrijeme žele

prijeći na *proof of stake* algoritam. *Proof of stake* algoritam možemo opisati kao algoritam koji kao dokaz uzima ulog u tu kriptovalutu, te se po količini kriptovalute koje posjedujemo odabire tko će biti odabran za validaciju novog bloka u *blockchainu*. Remix IDE besplatni je online alat koji služi za testiranje koda i kreiranje pametnih ugovora.

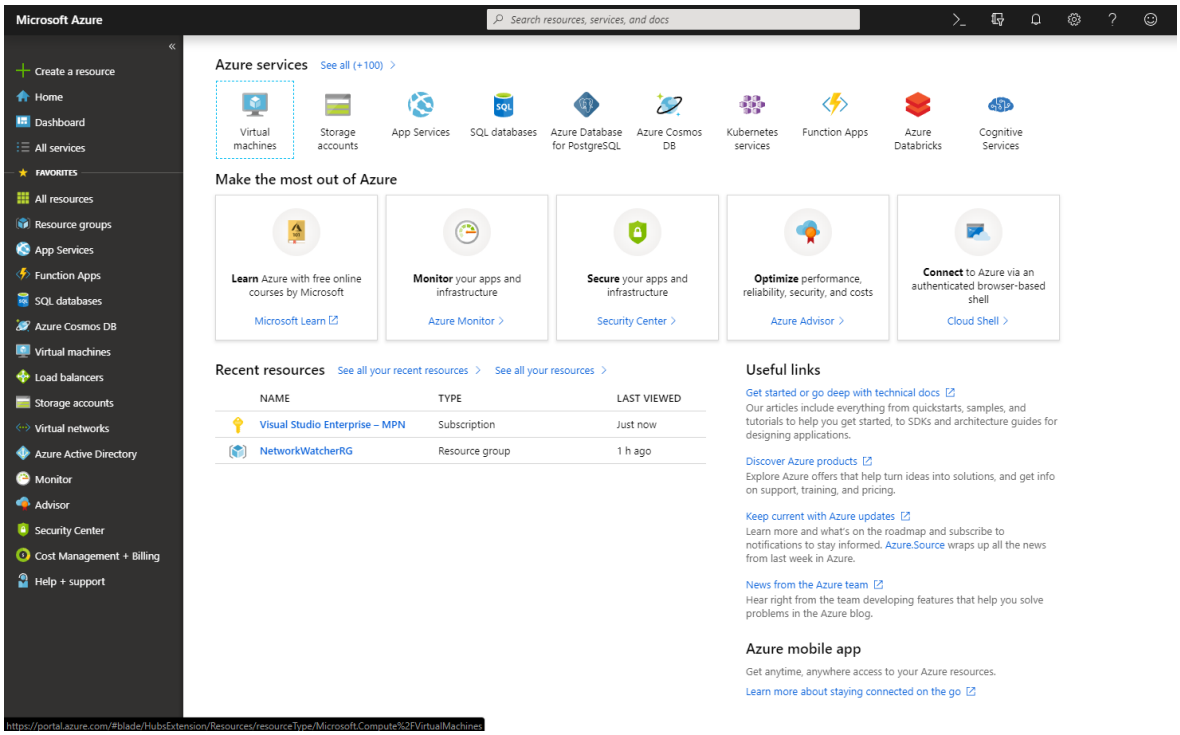
5. Primjena blockchain tehnologije u prehrambenoj industriji

5.1. Implementacija blockchaina

Microsoft Azure

Microsoft Azure je usluga računarstva u oblaku. Umjesto pokretanja vlastitog hardvera ili plaćanje određenog hardvera u tuđem podatkovnom centru, plaćamo samo pristup ogromnom skupu računalnih resursa koje nudi Microsoft. To nam omogućuje da kreiramo *web* poslužitelje, poslužitelje za e-poštu, baze podataka, poslužitelje za pohranu datoteka, virtualne strojeve, korisničke direktorije ili bilo što drugo što se nudi na njihovom Azure Portalu [15]. U slučaju kada nam je potrebno proširenje računalnih resursa, ne moramo kupovati fizički hardver komponente, već oblak (engl. *Cloud*) automatski raspoređuje resurse. Plaćamo samo onoliko računalnih resursa koliko nam je potrebno. Takve usluge mogu biti ljavni poslužitelj dostupan svima ili dio privatnog poslužitelja koji se koristi samo u našoj organizaciji. Pruža usluge softver kao uslugu (engl. *Software as a Service*), platformu kao uslugu (engl. *Platform as a Service*) i infrastrukturu kao uslugu (engl. *Infrastructure as a Service),* te podržava mnoge različite programske jezike, alate i okvire.

Glavna prednost ovakve usluge je manji trošak pri korištenju računalnih resursa. Ne moramo ulagati puno novca u stvaranje vlastitog podatkovnog centra, kupnju hardvera za njega i plaćanje ljudi koji će održavati isti. Plaćamo samo ono što koristimo i to koliko koristimo.



Slika 3. Microsoft Azure portal

Neke od usluga Microsoft Azure-a su:

- Virtualne mašine
- Mobilne usluge
- Usluge skladištenja podataka
- Usluge upravljanja podacima
- Usluge razmjena poruka
- Razne metrike
- Strojno učenje
- Funkcije
- Internet stvari

Blockchain as a Service

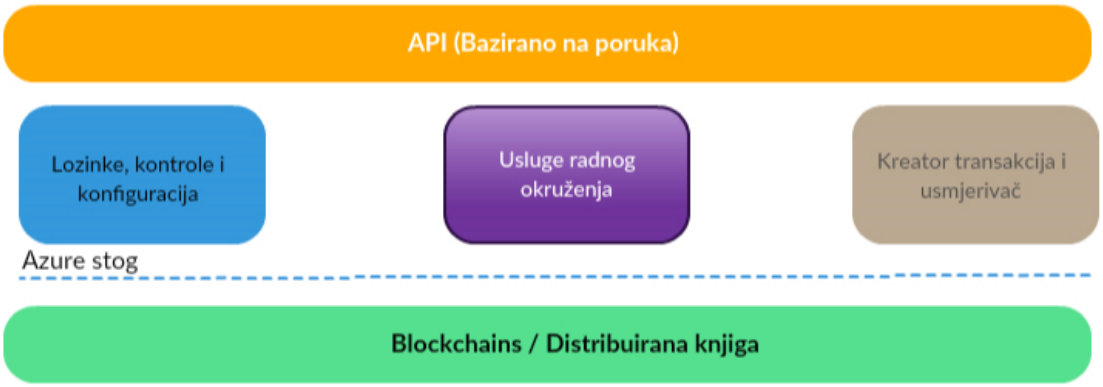
Prilikom izrade praktičnog dijela ovog rada koristio sam Microsoft Azure uslugu Blockchain as a Service. Blockchain na Azure-u pruža brzu, jeftinu platformu niskog rizika za izgradnju i implementaciju *blockchain*-a. Blockchain as a Service na Azure-u u osnovi nudi nekoliko različitih predložaka za implementaciju spremnih za korištenje čak i u rješenjima velikih poduzeća, a neke od njih su:

- BlockApps STRATO
- Ethereum
- Quorum
- Hyperledger Fabric

Ključne mogućnosti Azure Blockchain as a Service platforme

- Pojedinačni čvor (engl. *Single-node)* glavne knjige koji simulira produkciju za više odjela unutar jedne organizacije
- Višestruki čvor (engl. *Multi-node*) glavne knjige koji simulira produkciju za više odjela unutar više organizacija
- Alate za razvoj decentraliziranih aplikacija baziranih na *blockchainu*

Decentralizirane aplikacije (dApps) su aplikacije koje se izvode na ravnopravnoj mreži računala umjesto na jednom računalu. Zamislimo dApp kao klijentsku aplikaciju koja komunicira s pametnim ugovorom za interakciju s *blockchain* mrežom.



Slika 4. Azure Blockchain as a Service arhitektura

Ključne značajke izgradnje *blockchaina* na Azuru su:

- Uspostavljane sigurnog okruženja putem Azure Virtual Networks, Azure App Services VNet integracije ili mrežne sigurnosne grupe
- Izrada pametnih ugovora koristeći neki od dostupnih razvojnih alata kao što su Blockstack Core, Ethereum Stdio ili Truffle

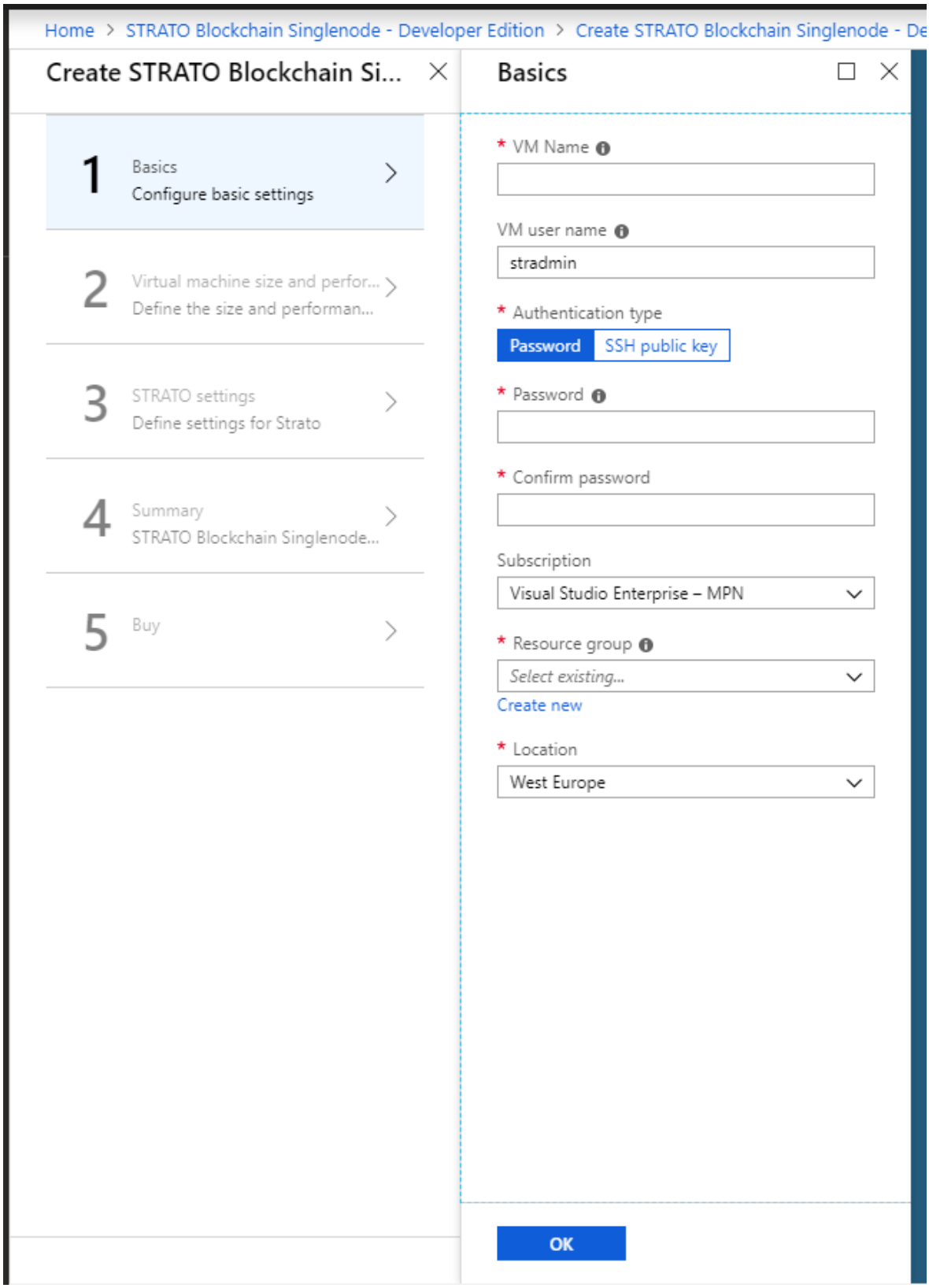
- Automatiziranje implementacije komponenti sudionika, virtualnih strojeva i komponenti PaaS koje omogućuju Azure Resource Manager i Powershell skripte
- Zaštita pristupa podacima i logici implementacijom Azure AD u aplikacije i API-e
- Izgradnja arhitekture za integraciju poslovnih rješenja korištenjem Azure-a za poduzeća i distribuciju širom svijeta

Koraci kreiranja STRATO Blockchain as a Service na Azure -u:

Prvo unosimo osnovne podatke virtualne mašine na kojoj će biti postavljen naš *blockchain.*

1. korak:

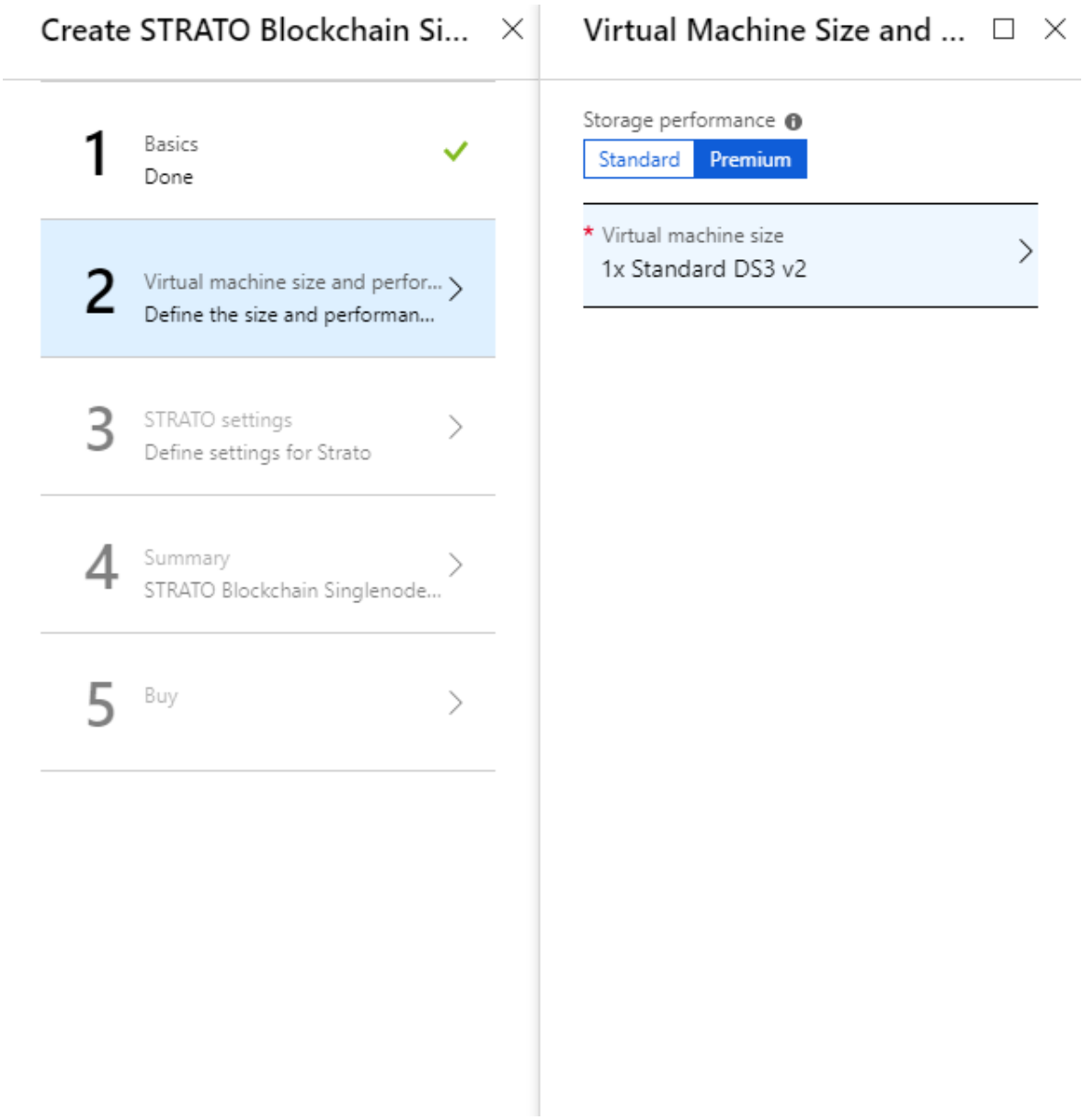
Naziv virtualne mašine i naziv korisnika unosimo proizvoljno. Nakon toga odabiremo način autorizacije na virtualnu mašinu, možemo birati između lozinke i SSH *public key*-a. U mom slučaju odabrao sam prvi način autorizacije. Slijedeće po redu je *Subscription* tj. pretplata, koju odaberemo sukladno pretplati na koju je pretplaćen naš korisnički račun. Potom odabiremo grupu resursa (engl. *Resource group)* u koju će biti pohranjeno sve što nam je potrebno za naš *blockchain,* a to su diskovi, virtualna mašina (ili više njih), mrežna sigurnosna grupa, mrežni *interface,* ip adresa i virtualna mreža. Na posljetku odabiremo lokaciju, važno je odabrati dobru lokaciju zbog cijene koja varira ovisno o području u kojem se nalazite.



Slika 5. Unos osnovnih podataka

2. korak:

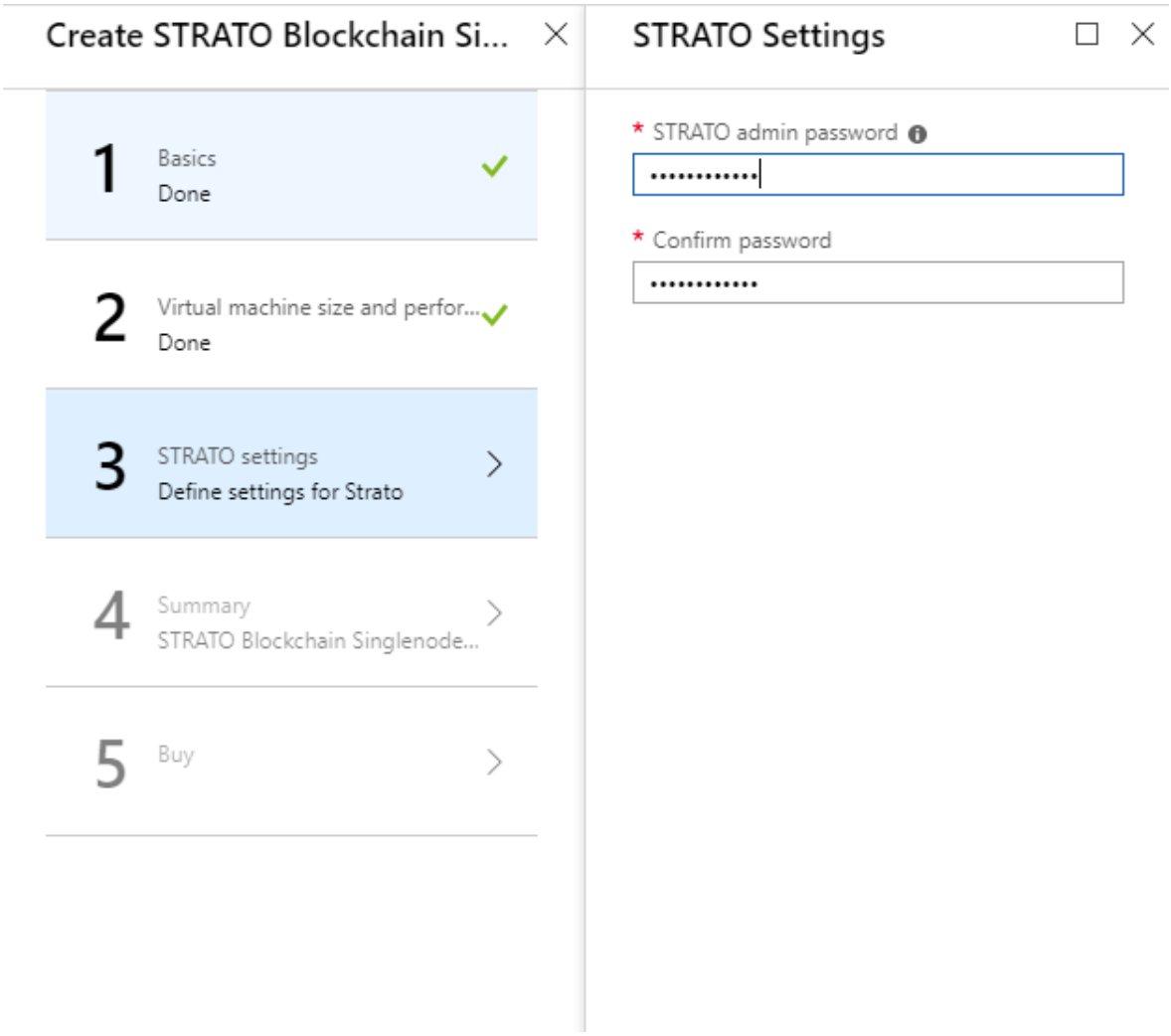
U drugom koraku odabiremo veličinu i performanse virtualne mašine. Standard performanse su jeftinije ali i lošije od Premium performansi, mašina koristi hard disk koji je sporiji te ima manje memorije nego u Premium paketu koji koristi SSD disk.



Slika 6. Odabir performansa virtualne mašine

3. korak:

U trećem koraku odabiremo STRATO admin lozinku za pristup blockchainu

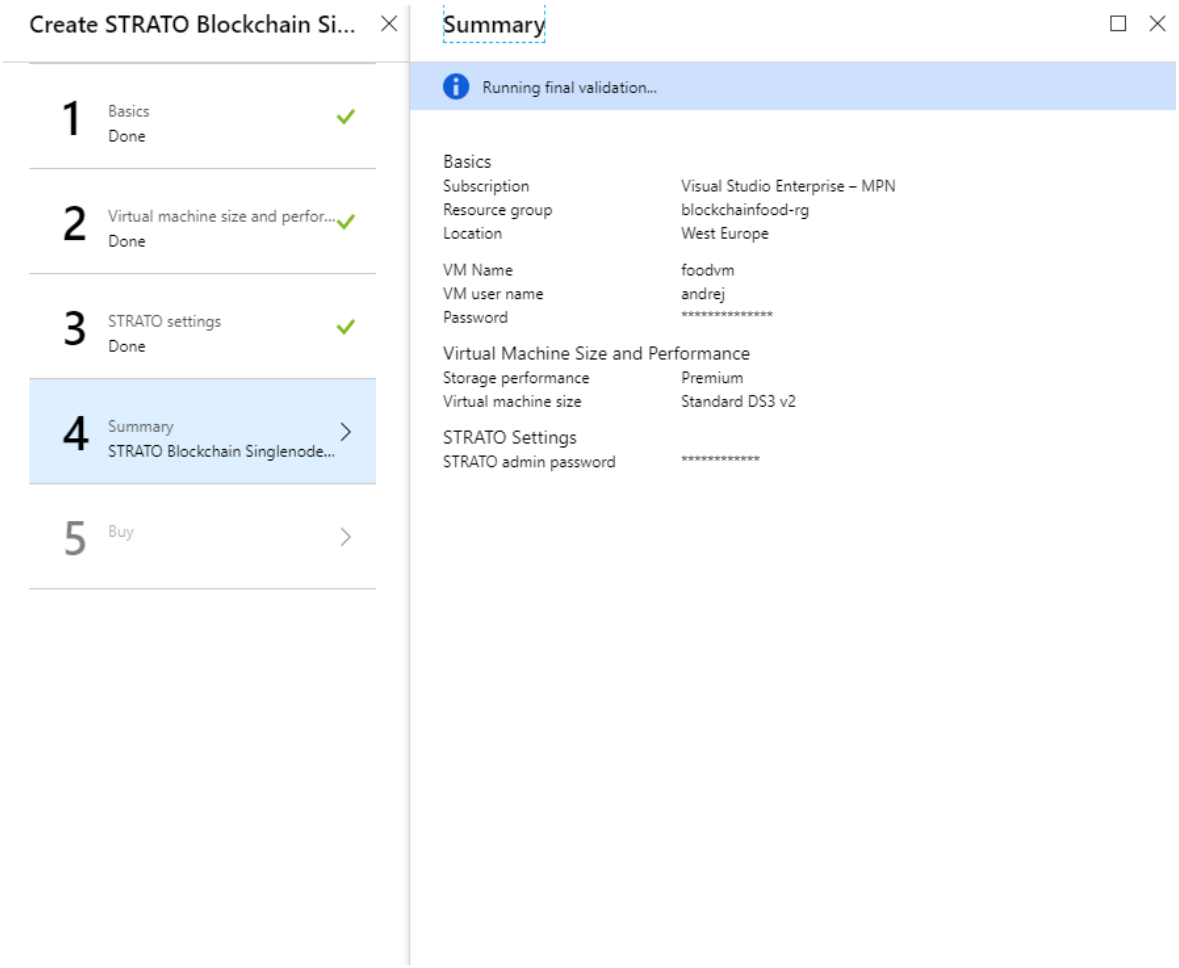


Slika 7. Odabir administratorske lozinke

.

4. korak :

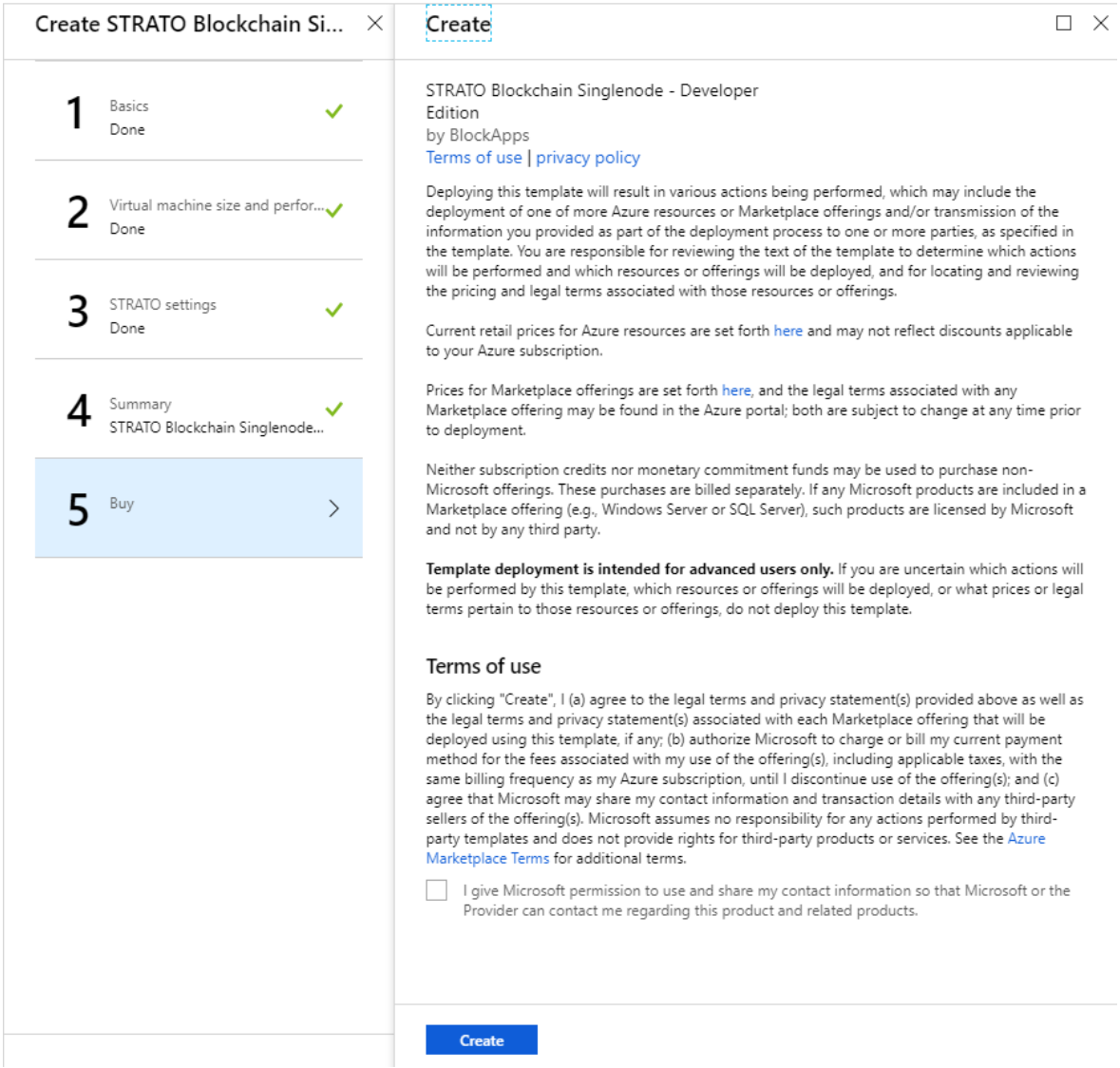
Odrađuju se zadnje validacije, prikazuju se pojedinosti koje smo odabrali u prošlim koracima.



Slika 8. Validacije

5. korak:

U ovom koraku dobijemo objašnjenje od Microsofta što smo sve dobili kreiranjem ovog *blockchaina*, te neka druga korisnička pravila i upute. Pritiskom na tipku *Create* naš *blockchcain* se kreira, te nakon toga je spreman za korištenje.



Slika 9. Objašnjenje od Microsofta kao zadnji korak

BlockApps STRATO

U nastavku ću opisati BlockApps STRATO Blockchain as a Service koji sam implementirao u praktično rješenje završnog rada. BlockApps STRATO je rješenje za brzu implementaciju *blockchain*-a za bilo koju primjenu.

Karakteristike BlockApps STRATO *blockchaina:*

- Integracija – jednostavno integriranje s bilo kojim *web* rješenjem i bazama podatka koristeći REST API
- Upravljanje – provjera pristupa API-u i zaštita svojih podataka
- Pretraživanje – omogućuje pretraživanje milijune pametnih ugovora u sekundi pomoću uobičajenih jednostavnih SQL upita
- Zaštita – razmjenjujemo podatke samo sa sudionicima *blockchaina* kojima smo mi odobrili pristup, te transakcije između nas su vidljive samo nama
- Prijenos – mogućnost prijenos pametnih ugovora i podataka u blockchain kako bi pokrenuli novi projekt
- Implementacija – Microsft Azure, AWS i Google Cloud omogućuje postavljanje i hostanje STRATO *blockchaina*

BlockApps je stvorio implementaciju Enterprise Ethereum protokola, zajedno s alatima i API-jem za interakciju sa podatcima na *blockchainu*.

Komponente koje čine STRATO platformu:

- EVM – Ethereum virtualna mašina koja izvršava pametne ugovore, koji su napisani u programskom jeziku Solidity, te se kod kompajlira Solidity kompajlerom u bajtni kod koji je razumljiv EVM
- P2P – ova komponenta komunicira sa ostalim STRATO čvorovima, prenosi podatke i transakcije s drugim sudionicima
- Kafka – interna sabirnica poruka koju koriste različite komponente STRATO platforme
- REST API - različiti API-ji omogućavaju programerima interakciju s STRATO platformom, što im omogućuje bolje obavljanje raznih zadataka
- Analitičke baze podataka – informacije iz *blockchaina* indeksirane zbog lakšeg pronalaženja i dohvaćanja podataka
- Upravljačka ploča (engl. *Dashboard)* – korisničko sučelje za vizualnu interakciju i upravljanje STRATO čvorovima i mrežama [13]

STRATO Management Dashboard

U gornjem desnom kutu možemo primijetiti verziju trenutnog STRATO Management Dashboarda, ali osim toga nude nam se i dva API-ja. *Bloc API* upravlja svim podacima izvan lanca koji su potrebni za interakciju s STRATO *blockchain* -om. Ovo je API koji najčešće vežemo sa klijentskom aplikacijom. Upravlja privatnim ključevima, metapodacima ugovora, prati statuse transakcija, te je svjestan stanja ugovora. STRATO API koristi se za interakciju sa *blockchainom* i podacima na njemu.

STRATO API nam omogućuje:

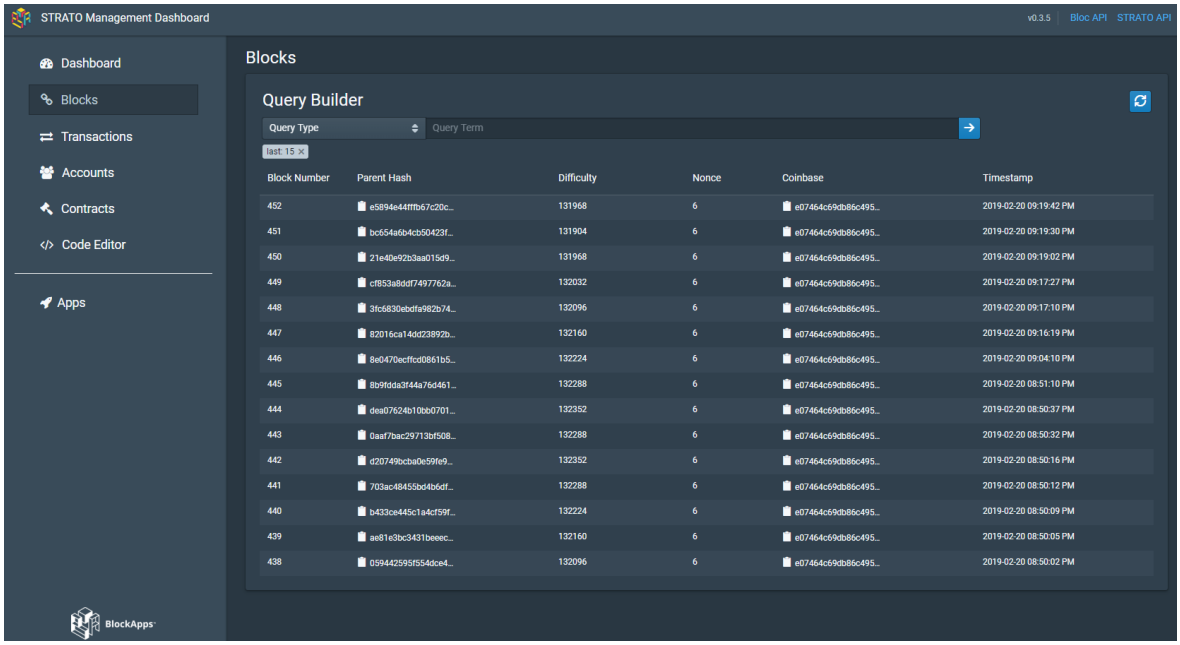
- Traženje informacija o bloku
- Traženje transakcija
- Traženje podataka o računu
- Slanje potpisanih transakcija
- Provjeru transakcija
- Provjeru informaciju o računima

Dashboard

Slika 10. Pregled *Dashboard*

Pregled *Dashboard* omogućava uvid u stanje našeg čvora. Prikazuje nam stanje mreže, koliko i koji blok je zadnji dodan u ovom čvoru, koliko je korisnika trenutno smješteno na ovom čvoru, te koliko je pametnih ugovora postavljeno na ovaj čvor. Ispod prvog vertikalnog prikaza, nalaze se metrike. Prva od njih je težinska vrijednost (engl. *Difficulty)* koja nam govori kolika je trenutna težina za dodavanje novog bloka. Slijedeća u nizu metrika je ukupni broj obavljenih transakcija (engl. *Transaction Count).* Nakon nje dolazi nam vrijeme potrebno za dodavanje novog bloka (engl. *Block Propagation)* koji se mjeri u sekundama. Tip transakcije (engl. *Transaction Type)* prikazuje nam na dinamičnom grafu, koje vrste transakcija su dodane, a neke od vrsta su ugovori (engl. *Contract),* transferi (engl. *Transfer),* te poziv funkcije (engl. *Function call).* U slijedećem retku na prvom mjestu nalaze se podaci o čvoru kao što su hash vrijednost čvora, broj blokova, te broj ravnopravnih partnera za taj čvor. S desne strane nalazi se popis posljednjih transakcija u čvoru kao dinamičan prikaz, klikom na redak otvaraju se pojedinosti označene transakcije.

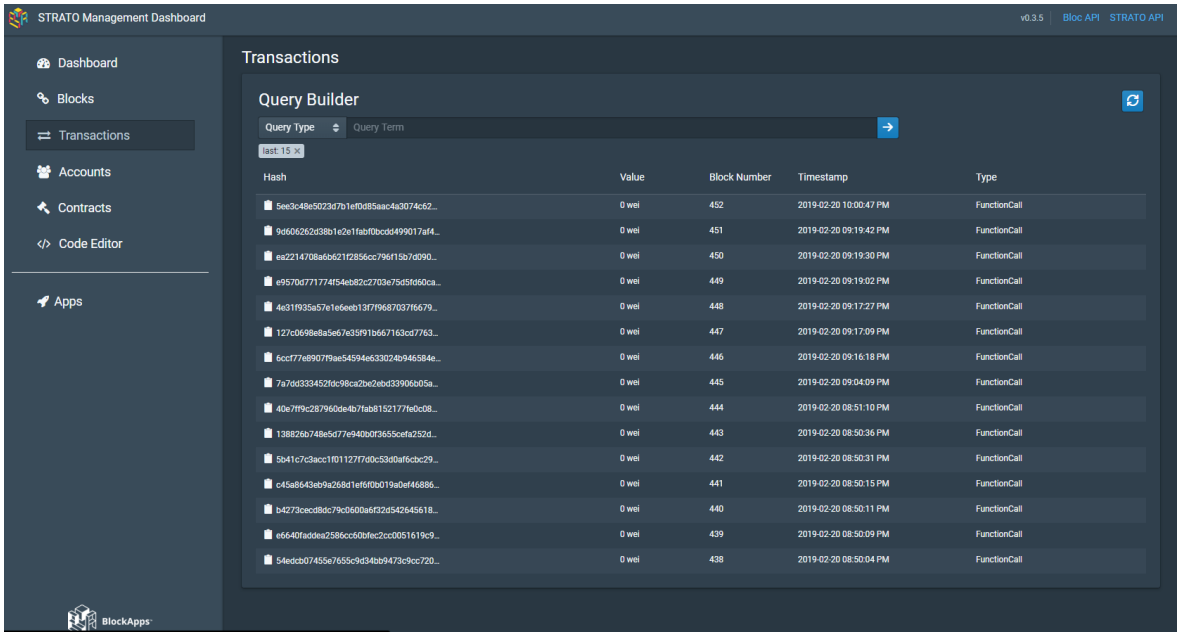
Blocks



Slika 11. Pregled *Blocks*

Slijedeći po redu u okomitom meniju nalazi se *Blocks.* Pregled koji nam omogućuje pretraživanje *blockchaina*. Alat kroz korisničko sučelje omogućava sql upite prema *blockchainu.* Kombinacijom predodređenih tipova upita (engl. *Query Type)* u padajućem izborniku s lijeve strane pregleda i upitnog termina (engl. *Query Term)* šaljemo upit prema *blockchainu,* a kao rezultat dobivamo sve blokove koji odgovaraju zadanim parametrima. Klikom na pojedini redak, prikazuju nam se detalji bloka kao zasebni prikaz.

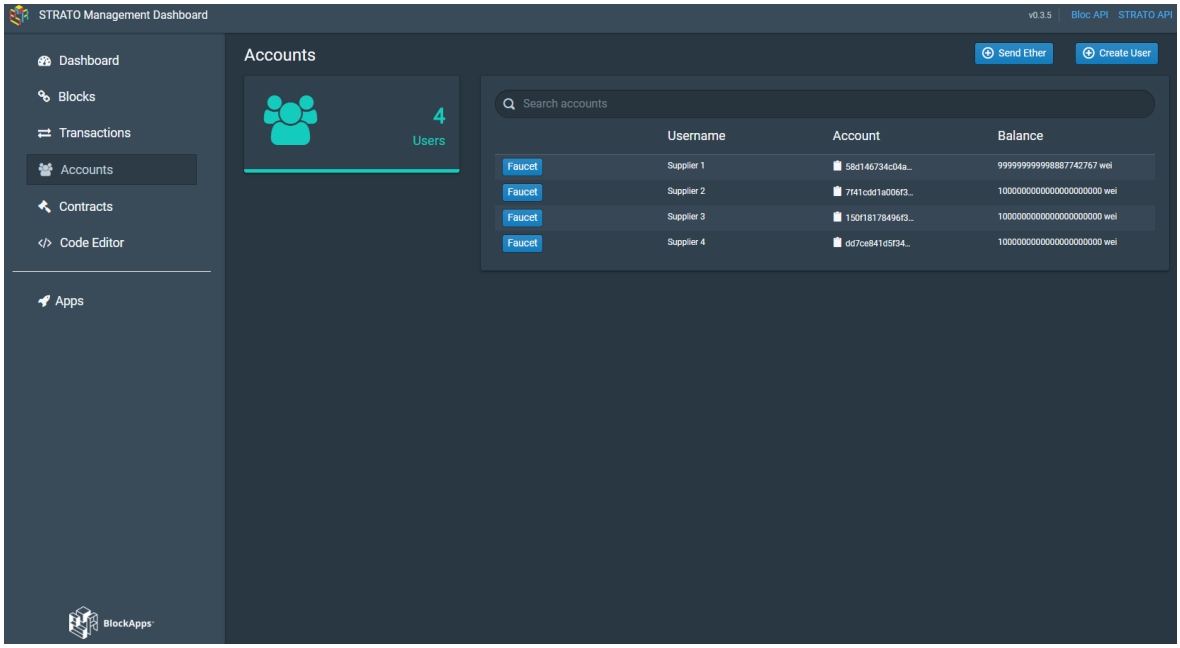
Transactions



Slika 12. Pregled *Transactions*

Nakon *Blocks*-a dolazi nam *Transactions* pregled. U principu pregled je po funkcionalnostima jednak prijašnjem pregledu, no u ovom pregledu pišemo upite za transakcije u blokovima, također kombinacijom predodređenog tipa upita i upitnog termina.

Accounts



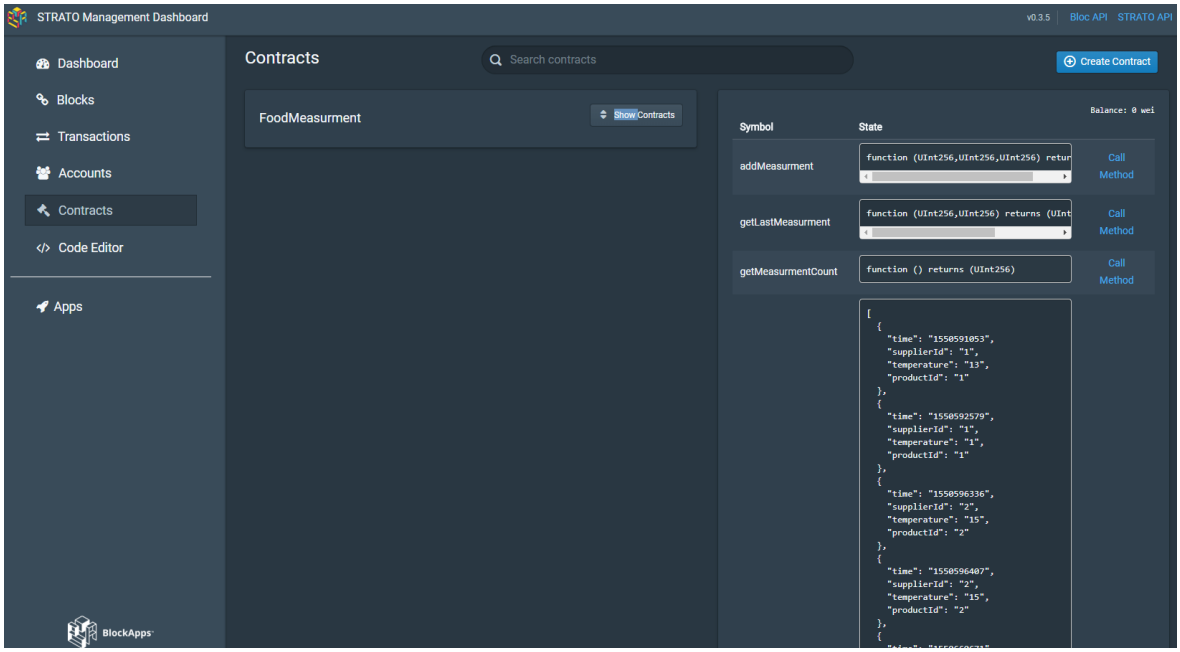
Slika 13. Pregled *Accounts*

Pregled *Accounts* prikazuje sve račune na našem *blockchainu*. Klikom na bilo koji od računa prikazuju se detalji tog računa kao što su *hash* vrijednost računa (engl. *Contact Root),* vrsta računa (engl. *Kind),* iznos na tom računu (engl.*Balance),* broj zadnjeg bloka (engl. *Latest Block Number),* kodni *hash* računa te broj *nonce.*

Osim navedenih detalja u desnom kutu ekrana nalaze se dva gumba. Pritiskom na prvi gumb *„Send Ether“* moguće je poslati određeni iznos Ethereuma drugom računu. Otvara se dodatni pregled na kojem se odabire s kojeg računa želimo slati, s koje njegove adrese, lozinka, kome šaljemo, na koju adresu šaljemo, te iznos koji želimo poslati.

Sljedeći gumb „*Create User“* omogućava nam dodavanje novog korisničkog računa za ovaj čvor.

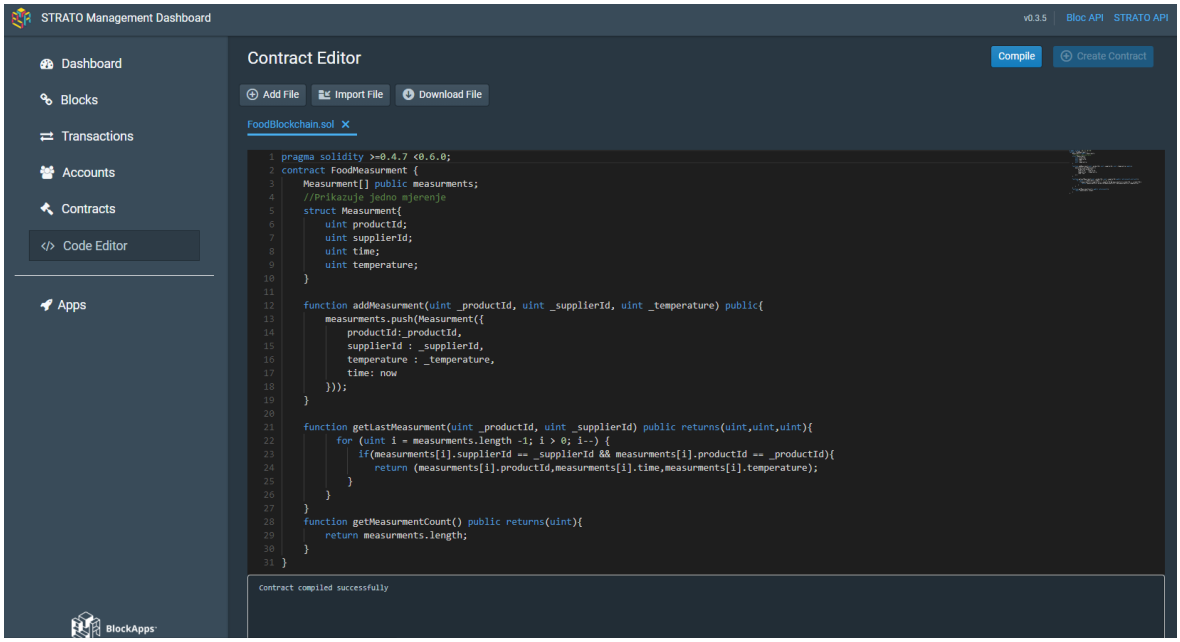
Contracts



Slika 14. Pregled *Contracts*

Pregled *Contracts* omogućuje nam pregled svih ugovora na *blockchainu.* Za svaki ugovor koji se nalazi na *blockchcainu* omogućava pregled stanja određenog ugovora, te prethodno prenesene verzije istog ugovora. Pritiskom na gumb „*Create Contract*“ omogućuje nam se kreiranje novog ugovora ili prijenos već postojećeg. S obzirom da se ugovor može pisati na u više različitih integriranih razvojnih okruženja, BlockApps STRATO omogućava prijenos .sol datoteke koja sadržava pametni ugovor. Svakom ugovoru također je moguće poslati personalizirane upite kako bi ih testirali.

Code Editor



Slika 15. Pregled *Code Editor*

Code Editor omogućuje pisanje pametnih ugovora. Postoji mogućnost dodavanja datoteka, uvoza datoteka i preuzimanja datoteka. Koristi kompajler koji provjerava valjanost napisanog ugovora, te u dnu prozora možemo vidjeti izlazni rezultat kompajliranja.

5.2. Implementacija pametnih ugovora

Pametni ugovori su samo izvršavajući ugovori sklopljeni između dviju ili više strana bez potrebe za središnjim autoritativnim tijelom. Izvršavaju se na računalima, te postaju valjani onda kada je određeno u ugovoru. Najpoznatiji i najbrojniji pametni ugovori pisani su u Solidity programskom jeziku.

Solidity je objektno orijentirani programski jezik za implementaciju pametnih ugovora. Kombinacija je C ++, Python i JavaScript programskih jezika, a dizajniran je da se izvršava na Ethereum Virtual Machine (EVM) [14]. Podržava nasljeđivanje (engl. *Inheritance*), knjižnice (engl. *Libraries)* i kompleksne korisnički definirane tipove.

Neki od područja korištenja pametnih ugovora su glasanje, aukcije, grupno financiranje, prehrambena industrija, bankarski sektor itd.

Prije no što krenemo sastavljati pametni ugovor, potrebno je uzeti u obzir sve pojedinosti koje želimo usuglasiti. Ova stavka je jako bitna zbog toga što jednom sklopljeni ugovor više ne može biti izmijenjen. Kada se ugovor postavi na *blockchain*, a recimo sadrži neku pogrešku, do trenutka kada ju otkrijemo jedna od strana može biti oštećena.

```
pragma solidity >=0.4.7 <0.6.0;
contract FoodMeasurment {
Measurment[] public measurments;
struct Measurment{
  uint productId;
  uint supplierId;
  uint time;
  uint temperature;
    }

function addMeasurment(uint _productId, uint _supplierId,
uint _temperature) public{
        measurments.push(Measurment({
            productId:_productId,
            supplierId : _supplierId,
            temperature : _temperature,
            time: now
        }));
    }

function getLastMeasurment(uint _productId, uint _supplierId)
public returns(uint,uint,uint){
for (uint i = measurments.length -1; i > 0; i--) {
if(measurments[i].supplierId == _supplierId &&
measurments[i].productId == _productId){
   return
(measurments[i].productId,measurments[i].time,measurments[i].
temperature);
            }
        }
    }
    function getMeasurmentCount() public returns(uint){
        return measurments.length;
    }
}
```

Kôd 1. Pametni ugovor napisan u Solidity programskom jeziku

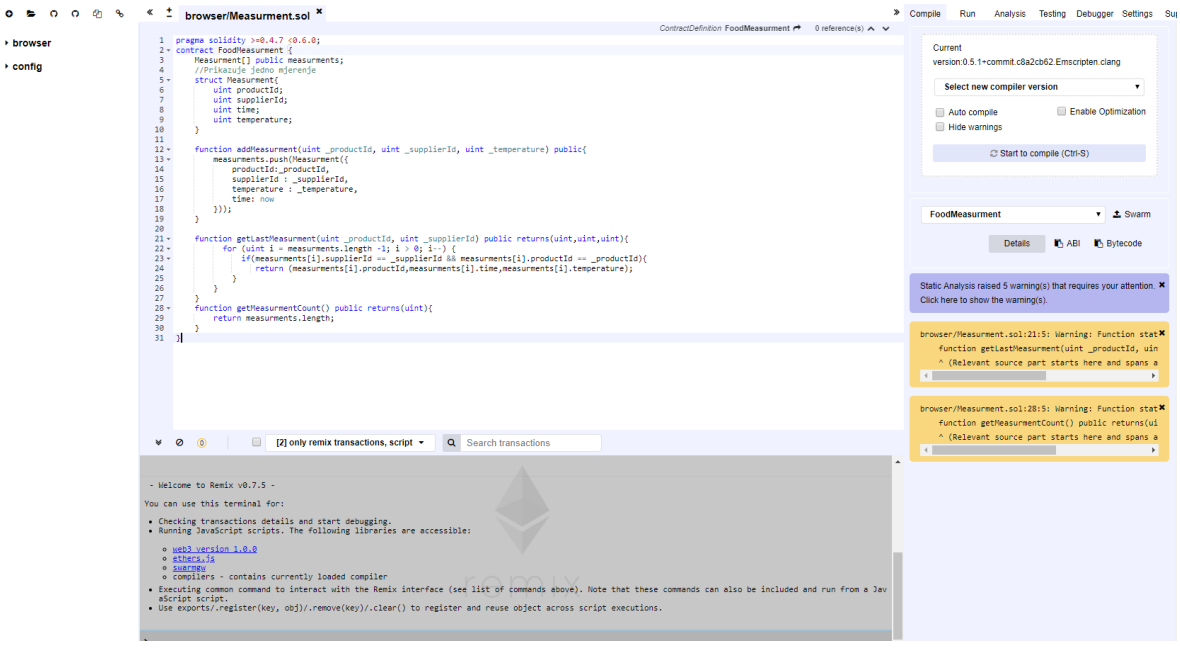
Prva linija nam govori da je ovaj kod pisan u Solidity verziji 0.4.7 ili nekoj novijoj, ali ne uključujući verziju 0.6.0. Ovaj dio nam govori s kojim kompajlerima je kompatibilan ovaj kod. Inače, pragma znači uobičajene upute za kompajlere o tome kako tretirati izvorni kod.

Riječ ugovor (engl. *Contract)* označava početak pametnog ugovora. Ugovor je skup koda (njegovih funkcija) i podataka (njegovog stanja). Riječ struktura (engl. *Struct)* označava jedan korisnički definirani tip objekta koji se definira i kasnije koristi kao bilo koji drugi tip. Uint označava varijablu tipa *unsigned integer* od 256 bitova. Uglate zagrade označavaju da se radi o polju, a tip koje će polje sadržavati definirano je prije zagrada.

Function je kao i u drugim programskim jezicima oznaka za funkcionalnost, u ovom slučaju ta je funkcionalnost opisana u ugovoru tako da je to funkcionalnost ugovora. Obavezno je navesti naziv funkcije, te dostupnost iste riječima *public* ili *private.* Postoji mogućnost definiranja parametara koja će sama funkcija primati. Nakon definiranja dostupnosti, moguće je definirati i tip i vrstu objekta koji će biti vraćen kao rezultat radnje koju obavlja funkcija, za to se koristi riječ *return*.

Za pisanje koda moguće je lokalno na računa postaviti kompajler koda, no postoji i nekoliko *online* integriranih razvojnih okruženja (engl. *IDE* – Integrated Development Environment).

Za slučaj pisanja pametnog ugovora u praktičnom dijelu rada koristio sam Remix IDE.



Slika 16. Sučelje Remix IDE

Remix IDE je *open source* alat koji nam pomaže da napišemo ugovore u programskom jeziku Solidity izravno iz preglednika. U njemu su sadržani skupovi alata za interakciju sa Ethereum *blockchain* -om. Napisan je u programskom jeziku JavaScript.

Remix podržava testiranje, otklanjanje pogrešaka i implementaciju pametnih ugovora.

Ugovor napisan za praćenje temperature hrane u svim entitetima lanca prikazan je u kodu Kôd 1. Napravio sam ugovor pod nazivom `FoodMeasurment` koji u sebi sadrži strukturu pod nazivom mjerenje (engl. *Measurment).* `Measurment` u sebi sadrži četiri parametra, a to su `productId` jedinstvena oznaka proizvoda, `supplierId` jedinstvena oznaka dobavljača, `temperature` izmjerena kod dobavljača, te `time` kada je došlo do zabilježene temperature. Ugovor sadrži dvije funkcije. Prva se zove `addMeasurment` koja prima parametre `productId`, `supplierId` i `temperature`, a za vrijednost `time` postavlja trenutno vrijeme. Novi *Measurment* koji se kreira sa ovim parametrima dodaje se u dinamično polje `Measurments`[], te se pohranjuju na *blockchain.*

Druga funkcija pod nazivom `getLastMeasurment` kao parametar prima `supplierId` i `productId`, te vraća zadnje mjerenje za tog dobavljača.

Treća funkcija pod nazivom `getMeasurmentCount` vraća zbroj svih mjerenja u lancu.

5.3. Implementacija klijentske aplikacije za prikaz podataka sa blockchaina

Klijentska aplikacija omogućava dodavanje novog mjerenja, te prikaz zadnjeg mjerenja temperature kod nekog od dobavljača za određeni proizvod. Nakon što sam postavio pametni ugovor kroz STRATO korisničko sučelje, dobio sam dodatnu mogućnost poziva funkcija koje su definirane tim pametnim ugovorom. STRATO Rest API pušta prema van adresu preko koje se onda mogu vršiti pozivi direktno na *blockchain.* STRATO bloc API ima predefinirane adrese koje se popunjavaju određenim parametrima i na taj način se strukturira poziv prema *blockchainu.*

Slika 17. Prikaz ukupnog broja mjerenja i forma za unos novog mjerenja

Prvi dio klijentske aplikacije prikazuje ukupan broj zapisanih mjerenja na *blockchainu.* Ispod toga nudi se mogućnost unosa novog mjerenja za određeni proizvod i određenog dobavljača. Klikom na gumb potvrdi, postavili smo novo mjerenje sa zadanim parametrima.

Adresa koja se koristi u svim pozivima metoda pametnog ugovora je definirana konstantnom `url`, a u kodu izgleda :

```
const url =
'http://foodqogptp.westeurope.cloudapp.azure.com/bloc/v2.2/us
ers/' + userName + '/' + userAddress + '/contract/' +
contractName + '/' + contractAddress + '/call?resolve';
```

Kao što možemo vidjeti, prvi dio adrese je adresa virtualne mašine kreirane na Azure-u. Od parametara koji su nužni u adresi dodaju se naziv korisnika (u našem slučaju naziv dobavljača), njegova *hash* adresa koju pronalazimo u pregledu „*Accounts"* u STRATO korisničkom sučelju klikom na naziv korisnika, naziv pametnog ugovora i njegova *hash* adresa koju pronalazimo u pregledu „*Contracts*" također u STRATO korisničkom sučelju.

Naziv metode za dohvaćanje svih mjerenja je `getMeasurmentCount`.

Prilikom kreiranja poziva, u React-u pozivamo metodu `fetch` koja za parametre prima url, tip REST poziva (POST), zaglavlja, tijelo poruke, te se kasnije rezultati dobiveni kao odgovor koriste i prikazuju na sučelju.

```
fetch(
      url,
      {
        method: 'POST',
        headers: {
          'Accept': 'application/json',
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          password: password,
          method: "getMeasurmentCount",
          value: 0,
          args: {}
        })
      })
      .then(res => res.json())
      .then(
        (result) => {
          this.setState({
            isLoaded: true,
            items: result
          });
        },
        (error) => {
          this.setState({
            isLoaded: true,
            error
          });
        }
      )
```

Kôd 2. Poziv za dohvaćanje svih mjerenja zapisanih na blockchainu

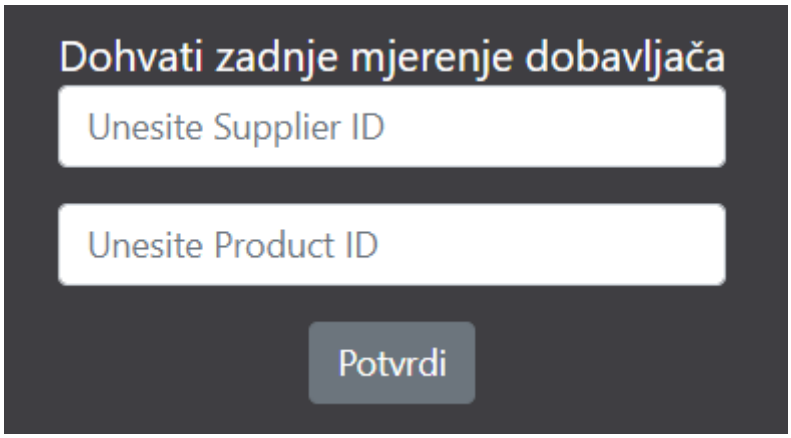
Naziv metode za unos novog mjerenja je `addMeasurment`. Poziv metode je vrlo sličan pozivu metode `getMeasurmentCount`, a razlikuje se u tijelu poziva.

```
fetch(
    url,
    {
      method: 'POST',
      headers: {
        'Accept': 'application/json',
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        password: password,
        method: "addMeasurment",
        value: 0,
        args: { _supplierId: supIDnm, _productId: prodIDnm,
_temperature: temp }
      })
    })
    .then(res => res.json())
    .then(
      (result) => {
        return "Mjerenje uspješno uneseno!";
      },
      (error) => {
        this.setState({
          isLoaded: true,
          error
        });
      }
    )
```

Kôd 3. Poziv za dodavanje novog mjerenja na blockchain

U tijelu REST poziva, postavljaju se parametri uneseni u korisničkom sučelju klijentske aplikacije. Parametri koji se unose su `supplierId`, `productId` i `temperature`. Kada se mjerenje unese ispravno na korisničkom sučelju prikazuje se poruka „Mjerenje uspješno uneseno!“, dok se u slučaju greške prikazuje poruka greške.

Drugi dio klijentske aplikacije omogućava dohvaćanje podataka o zadnjem mjerenju nekog od dobavljača za određeni proizvod.



Slika 18. Prikaz forme za dohvat zadnjeg mjerenja dobavljača za određeni proizvod

Unosom parametara `supplierId` i `productId`, te klikom na gumb „Potvrdi“ upućujemo poziv prema *blockchainu*.

U kodu klijentske aplikacije obavlja se sljedeći poziv

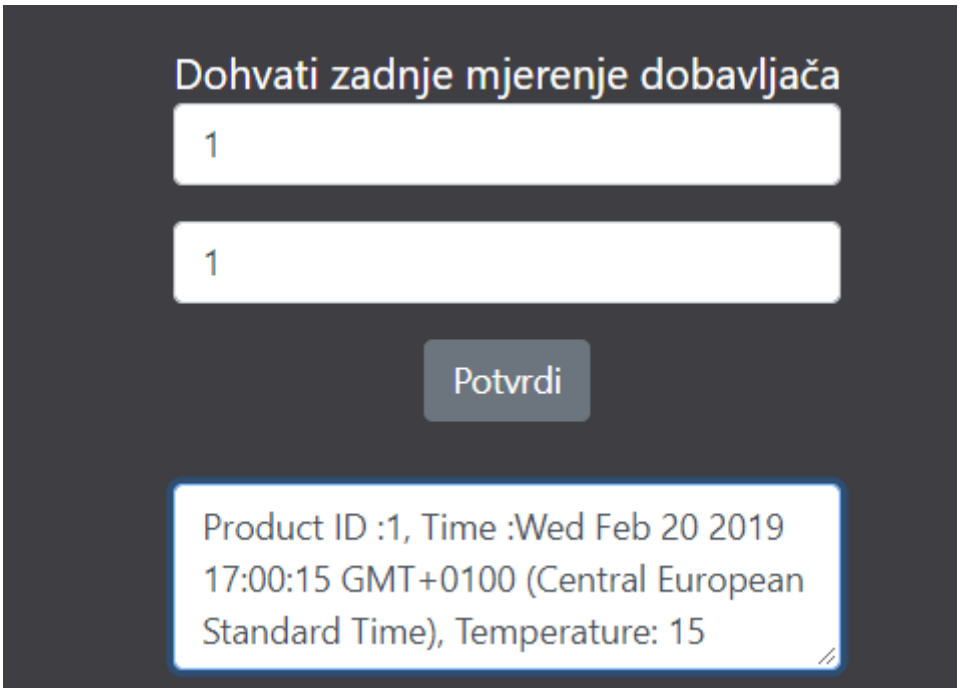
```
fetch(
    url,
    {
      method: 'POST',
      headers: {
        'Accept': 'application/json',
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        password: password,
        method: "getLastMeasurment",
        value: 0,
        args: { _supplierId: supID, _productId: prodID }
      })
    })
    .then(res => res.json())
    .then(
      (result) => {
        prikaziZadnjeMjerenje(result)
      },
      (error) => {
        this.setState({
```

```
            isLoaded: true,
            error
          });
        }
    )
```

Kôd 4. Poziv za dodavanje novog mjerenja na blockchain

U tijelu metode postavljamo parametre koje korisnik definira u korisničkom sučelju, te sa tim parametrima pozivamo metodu `getLastMeasurment` definiranu u ugovoru.

Kao odgovor dobivamo `ProductID`, `Time` i `Temperature` zadnjeg zabilježenog mjerenja.



Slika 19. Prikaz rezultata dohvaćanja zadnjeg mjerenja

Predefinirane pozive koje omogućava STRATO Rest API, uvelike olakšavaju prikazivanje i manipulaciju podataka. Od REST poziva mogući su *GET* i *POST* tipovi poziva, dok *UPDATE* i *DELETE* nisu, iz razloga što jednom zapisani podaci na *blockchainu* nikada ne mogu biti promijenjeni ili obrisani.

Zaključak

U ovom radu obrađeni su osnovni koncepti i karakteristike *blockchain* tehnologije.

Teorijski su obrađeni dijelovi *blockchaina,* SHA-256 kriptografski algoritam, decentralizirani sustav, opisan je problem i rješenje problema usuglašavanja bizantskih generala, algoritmi koji se koriste u *blockchain* tehnologiji, te područja primjene *blockchain* tehnologije.

Blockchain tehnologija predstavlja distribuiranu listu digitalnih informacija podijeljenu između svih čvorova u lancu. *Blockchain* tehnologija nastala je 2008. godine, a njen je kreator pojedinac ili grupa ljudi predstavljajući se pod pseudonimom Satoshi Nakamoto. Osmišljena je za potrebe distribuiranja digitalnog novca Bitcoin. *Blockchain* predstavlja glavnu knjigu u kojoj su zapisane sve transakcije. Svaki čvor unutar *blockchaina* ima svoju kopiju knjige koja se ažurira svaki puta kada se nova transakcija zapiše u *blockchain*. *Blockchain* sustav je decentralizirani sustav, u kojem nema središnjeg autoriteta. Sustav je izgrađen po modelu ravnopravnih partnera (engl. *peer to peer*).

Postoje dvije vrste *blockchaina*, privatni i javni *blockchain*. Privatni *blockchain* se koristi u slučajevima kada informacije želimo dijeliti samo sa određenim ljudima, dok se javni koristi kada želimo da informacije zapisane u lancu budu javno dostupne. *Blockchain* pruža maksimalnu zaštićenost podataka korištenjem kriptografskih metoda.

U praktičnom dijelu rada prikazana je kompletna implementacija *blockchaina*. Korištenjem Microsoft Azure Blockchain as a Service usluge, te BlockApps STRATO Rest API aplikacije detaljno je prikazano rješenje spremno za korištenje u svim vrstama poduzeća bez obzira na veličinu poduzeća i poslovnu svrhu u koju će se koristiti *blockchain.*

Klijentski dio aplikacije prikazuje rezultate mjerena zapisane na *blockchainu,* te omogućuje unos novih zapisa mjerenja za određeni proizvod pri određenom entitetu u lancu.

Prilikom rada aplikacije prikazano je unošenje novih mjerenja za određeni proizvod i određeni entitet u lancu, zbroj dosadašnjih mjerenja na svim entitetima i za sve proizvode, te dohvaćanje zadnjeg mjerenja za određeni entitet i određeni proizvod.

Popis slika

Popis tablica

Popis kodova

Literatura

[1] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, (2008), https://bitcoin.org/bitcoin.pdf, 02.12.2018.

[2] https://www.solarwindsmsp.com/blog/centralized-vs-decentralized-network, 04.01.2019.

[3] https://medium.com/all-things-ledger/bitcoins-implementation-of-*blockchain*-2be713f662c2, 03.01.2019.

[4] What is hashing, https://blockgeeks.com/guides/what-is-*hash*ing/, 17.12.2018.

[5] What is the difference between decentralized and distributed systems?, Mari Edgar (2017), https://medium.com/distributed-economy/what-is-the-difference-between-decentralized-and-distributed-systems-f4190a5c6462, 15.01.2019.

[6] What is blockchain wallet?,https://blog.softwaremill.com/what-is-a-*blockchain*-wallet-bbb30fbf97f8, 17.01.2019.

[7] Ivan Voras, The Sceptic's guid to Bitcoin, Cryptocurrencies and the Blockchain, (2017)

[8] Dug Campbell, The Byzantine Generals Problem, (2015), http://www.dugcampbell.com/byzantine-generals-problem/, 14.12.2018.

[9] Shawn Gordon, What is Ripple?,(2017), https://bitcoinmagazine.com/guides/what-ripple/, 14.01.2019.

[10] Ethereum community, What is Ethereum?, (2016), http://ethdocs.org/en/latest/introduction/what-is-ethereum.html, 09.01.2019.

[11] Description of SHA-256, SHA-384, and SHA -512 , (2001), http://www.iwar.org.uk/comsec/resources/cipher/sha256-384-512.pdf, 05.01.2019.

[12] Wikipedia, SHA-2, https://en.wikipedia.org/wiki/SHA-2, 05.01.2019.

[13] BlockApps STRATO, https://blockapps.net/ , 20.12.2018.

[14] Solidity, https://solidity.readthedocs.io/en/v0.5.2/ , 24.01.2019

[15] Microsoft Azure, https://azure.microsoft.com/en-us/ , 13.01.2019.

[16] Merkel Tree, (2018), https://blockonomi.com/merkle-tree/, 09.01.2019.

[17] Making Sense of Blockchain Smart Contracts, https://www.coindesk.com/making-sense-smart-contracts, 07.01.2019.

[18] O'REILLY, Mastering Bitcoin – Early Release, https://unglueit-files.s3.amazonaws.com/ebf/05db7df4f31840f0a873d6ea14dcc28d.pdf 05.01.2019.