Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 007.52
DOI: https://doi.org/10.24867/09IH03Misic

IMPLEMENTACIJA PROGRAMSKE PODRŠKE ZA MANIPULACIJU PODACIMA DECENTRALIZOVANE APLIKACIJE ZA UPRAVLJANJE PRISTUPOM ZAŠTIĆENOJ ZONI

DATA MANIPULATION SOFTWARE IMPLEMENTATION OF DECENTRALISED APPLICATION FOR PROTECTED AREA ENTRY CONTROL

Živko Mišić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – MEHATRONIKA

Kratak sadržaj – *Potreba ljudi za sigurnošću uslovljava razvoj sigurnosnih sistema koji se primjenjuju za kontrolu ulaska osoba u određene prostorije. Iztraživanje se fokusira na mogućnosti primjene modernih tehnologija kako bi se razvio pouzdan i efikasan sigurnosni sistem. U okviru ovog rada, predstavlja se primjena Blokčejn tehnologije za razvoj decentralizovane aplikacije, kao i formiranje decentralizovane mreže računara koji upravljaju radom sigurnosnog sistema.*

Ključne riječi: *Blokčejn, Decentralizovana Aplikacija, Sigurnosni Sistemi.*

Abstract – *Human need for safety leads to constant development of security systems used for entry control of people in certain premises. Research is focused on possibility of applying modern technologies in order to develop reliable and efficient security system. This paper presents appliance of Blockchain technology for development of decentralised application and establishment of peer-to-peer network of computers operating the security system.*

Keywords: *Blockchain, Decentralised Application, Security Systems.*

1. UVOD

Sigurnost je oduvijek bila jedna od glavnih potreba ljudi, što se u jednoj velikoj mjeri ogleda u kontroli pristupa određenim posjedima, zgradama ili prostorijama. Sa navedenim potrebama i razvojem tehnologije, dolazi do mogućnosti razvoja pametnih vrata. Pametna vrata omogućavaju neprestanu kontrolu ulaska osoba čiji se identitet utvrđuje korištenjem njihovih biometrijskih karakteristika, što znači da se određenim osobama može dozvoliti odnosno zabraniti pristup, određenim prostorijama u određeno vrijeme.

Predmet istraživanja jesu mogućnosti primjene modernih tehnologija na primjeru realizacije sigurnosnog sistema pametnih vrata, čije se funkcionisanje zasniva na korištenju digitalnog ključa jedinstvenog za svakog korisnika, formiranog od njegovih biometrijskih podataka, odnosno od jedinstvenih karakteristika lica samog korisnika, te govorne fraze koja je takođe jedinstvena za svakog korisnika.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Bojan Mrazovac, docent.

Navedena funkcionalnost vrata zahtjeva istraživanja iz oblasti obrade zvuka, mašinske vizije i baza podataka. U okviru ovog rada, naglasak se stavlja na razvoj programske podrške za čuvanje podataka i formiranje mreže uz pomoć koje je moguće ostvariti komunikaciju između vrata, odnosno računarskih uređaja koji obrađuju i porede te podatke, te upravljaju radom vrata. S obzirom da je potrebno omogućiti unos i čitanje podataka sa više različitih mjesta, odnosno na više postojećih vrata, odabrana je Blokčejn tehnologija.

Neki od primjera sličnih sigurnosnih sistema:

- AFIS [1] - predstavlja sistem zasnovan na automatskoj identifikaciji otiska prsta. AFIS je najčešće korišten od strane zakonskih i izvršnih organa za identifikaciju kriminalnih osoba. Ovi sistemi se često primjenjuju i kao sigurnosni sistemi za potvrđivanje identiteta.
- ABIS [2] - predstavlja sistem automatske biometrijske identifikacije i on nadopunjuje AFIS sigurnosne sisteme. ABIS vrši poređenje trenutnog uzorka sa mnogim postojećim biometrijskim šablonima, koji se nalaze u bazi podataka, kako bi se pronašla individua i potvrdio njegov identitet. Pored otiska prsta, ABIS svoju sigurnost upotpunjuje prepoznavanjem biometrijskih karakteristika lica, očne zjenice i otiska dlana.

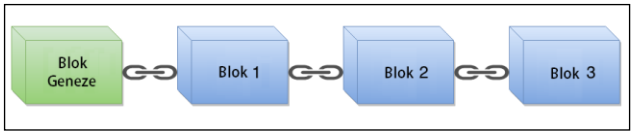
Iako već postoje mnoge realizacije sigurnosnih sistema koje koriste biometrijske karakteristike osobe, sistem koji je realiziovan u ovom radu je jedinstven usljed korištenja kombinacije prepoznavanja lica i govorne fraze za identifikaciju osobe, naravno to sve upotpunjeno korištenjem blokčejna za čuvanje i distribuciju podataka.

2. KORIŠTENE TEHNOLOGIJE

Za razvoj programske podrške za skladištenje i prikaz podataka projektovanog sigurnosnog sistema koriste se sledeće tehnologije:

2.1. Blokčejn

Blokčejn je tehnologija za čuvanje podataka. Može se zamisliti kao neprekidni niz, odnosno, lanac blokova prikazan na slici 1, odakle potiče i naziv (engl. *Block* što znači blok, te engl. *Chain* što znači lanac) [3]. Računari koji se povezuju na blokčejn nazivaju se čvorovima. Ovi čvorovi se međusobono povezuju korištenjem *peer-to-peer* protokola što predstavlja decentralizovanu mrežu gdje svaki čvor dijeli istu kopiju podataka, koja se takođe naziva digitalna sveska (engl. *Digital Ledger*).



Slika 1: *Slikovna prezentacija blokčejna*

Transakcijama je omogućena izmjena podataka po unaprijed definisanim pravilima. Izmjene se proslijeđuju svim čvorovima kako bi se ažurirala sopstvena kopija podataka. Nakon što je transakcija sačuvana i potvrđena od strane svih čvorova u mreži, više nije moguće promeniti podatke te transakcije. Svaki računar održava spisak transakcija u višestrukim uzastopnim blokovima gdje svaki blok pokazuje na prethodni blok. Proces potvrđivanja ovih transakcija se naziva rudarenje (engl. *mining*) i zasniva se na nekom od tipova konsenzusa koji predstavlja specifični algoritam na osnovu kojeg se postiže dogovor između čvorova pri usvajanju novog bloka. Ovi čvorovi koji vrše potvrđivanje transakcija se nazivaju rudarima (engl. *miner*).

U okviru ovog rada koristi se *Ethereum* [5] koji predstavlja danas najčešće korišten blokčejn okvir na tržištu, usljed čega postoji veliki broj primjera primjene i izvora za njegovo istraživanje kao i potrebnih programskih biblioteka za razvoj aplikacije, u odnosu na ostale blokčejn okvire fokusirane na razvoj aplikacija.

2.2. IPFS

Kada postoji potreba za čuvanjem određenih velikih podataka, koji zahtjevaju veće količine memorije kao što je slika, video itd., kod *Ethereum* blokčejna se javljaju izvjesne poteškoće [4]. Razlog za to jeste što je čuvanje ovih podataka zahtjevno a samim tim i preskupo, kako ogroman broj računara širom svijeta radi na potvrđivanju tih podataka ostvarivanjem dogovora. S obzirom da se novi blokovi podataka moraju potvrditi od strane velikog broja računara istovremeno, postoji ograničenje količine podataka koja može biti procesuirana u bloku. Kako bi se uklonili ovi problemi koristi se IPFS (engl. *Inter Planetary File System*). IPFS [6] je protokol i mreža dizajnirana korištenjem *peer-to-peer* protokola čuvanja i dijeljenja podataka. S obzirom da je prosječna količina podataka koja se postavlja u blok blokčejna oko 20 KB, što dovodi do visokih cijena ukoliko je potrebno sačuvati velike datoteke od više MB, IPFS predstavlja najbolje rješenje gdje je čuvanje i dijeljenje podataka brzo i jeftino. Prilikom pohrane podataka na IPFS, dolazi do generisanja jedinstvenog heša (engl. *Hash*) pomoću kojeg se pristupa podacima sačuvanim na IPFS-u. Heš predstavlja string znakova fiksne dužine, koji nastaje kao rezultat primjene matematičkog algoritma na podatak proizvoljne dužine. Dobijeni heš, koji je trenutno veličine svega 46 bajta, se zatim čuva na blokčejnu.

3. REALIZACIJA UPRAVLJANJA

Za realizaciju projekta, potrebno je razviti decentralizovanu aplikaciju za upravljanje pametnim vratima. Decentralizovana aplikacija (DApp) je termin koji se odnosi na programe programske podrške koji funkcionišu na *peer-to-peer* mreži računara zasnovanoj na blokčejnu. Za početak potrebno je formirati privatnu blokčejn i IPFS mrežu čije će čvorove činiti dvije razvojne ploče *Raspberry Pi 3 B+*[5] ili *Minnowboard Turbot B* [6] računara koji će predstavljati mozak pametnih vrata, kao i jedan dodatni računar na kojem će se vršiti obuka neuronske mreže za mašinsku viziju. Nakon umrežavanja računara, potrebno je napisati pametni ugovor u *Solidity* [3] programskom jeziku na osnovu kojeg funkcioniše decentralizovana aplikacija. Konkretno za dati slučaj, pametni ugovor je program na čijoj se adresi na blokčejnu nalazi baza podataka osoba koji su evidentirani u sigurnosni sistem, takođe unutar ugovora su definisana pravila pod kojim se može izvršiti transakcija određenih programskih funkcija, odnosno transakcija određenih podataka. I na kraju biće potrebna realizacija koda za čuvanje i preuzimanje datoteka korištenjem IPFS protokola.

3.1. Realizacija privatne Ethereum mreže

Kako bi se ostvarilo povezivanje računara u *Ethereum* blokčejn mrežu koristi se *Geth* koji predstavlja korisničku spregu za *Ethereum* komandnu liniju implementiran u *Go* programskom jeziku.

Geth omogućava pokretanje *Ethereum* čvorova na računaru. Na računaru se kreira novi direktorijum u kojem će biti smješteni podaci blokčejna kao i podaci o *Ethereum* nalogu. Prije pokretanja privatne blokčejn mreže, potrebno je formirati blok geneze.

Prvi blok u Blokčejn lancu, kao što je prikazano na slici 1, je poznat kao blok geneze (engl. *Genesis*). Blok geneze posjeduje bitne informacije koje se odnose na:

- Konfiguraciju blokčejn mreže,
- Alociranje početnih sredstava učesnika u mreži,
- Težinu usvajanja novog bloka,
- Broj za određivanje pravilnosti rudarenja,
- Maksimalnu količinu računanja koja se odnosi na rudarenje bloka,
- Vremensku oznaku formiranja bloka.

Nakon uspješne inicijalizacije bloka geneze, računari su spremni za pokretnje *Ethereum* blokčejn čvorova. Nakon uspješne konfiguracije čvorova slijedi umrežavanje svih potrebnih uređaja, nakon čega se pokreće rudarenje na željenim čvorovima.

3.2. Realizacija privatne IPFS mreže

Kako bi se omogućilo dijeljenje slika unešenih korisnika kao i kalsifikatora nakon obuke neuronske mreže, potrebno je uređaje povezati u privatnu IPFS mrežu, za čiju se realizaciju koristi *Go* implementacija IPFS-a.

Kako bi se IPFS čvorovi umrežili potrebno je da koriste istu *swarm.key* datoteku. Dakle, potrebno je na jednom računaru kreirati *swarm.key* korištenjem generatora *swarm* ključa, zatim tu istu *swarm.key* datoteku kopirati i prebaciti u sve direktorijume koji su kreirani prilikom inicijalizacije čvorova. IPFS zahtjeva postojanje jednog ili više *bootstrap* čvora preko kojeg se svi ostali čvori međusobno povezuju unutar mreže.

Bootstrap čvor je suštinski običan čvor sa dodatnom mogućnošću koja mu omogućava da povezuje klijentske čvorove privatne IPFS mreže. Nakon konfiguracije *bootstrap* čvora, potrebno je konfigurasti ostale čvorove u mreži što podrazumjeva unos IP4 adrese uređaja unutar konfiguracijske datoteke.

3.3. Realizacija Pametnog ugovora

Pametni ugovor predstavlja medium na osnovu kojeg će se ostvarivati međusobna interakcija korisnika i uređaja povezanih na Ethereum blokčejn mrežu. On će definisati koji se podaci čuvaju na blokčejn mreži, ko ima pravo pristupa tim podacima, odosno, pametni ugovor će definisati pravila na osnovu kojih funkcioniše cijela decentralizovana aplikacija. Za pisanje koda pametnog ugovora, odabran je *Solidity* programski jezik. Na osnovu razmatranja projektovanog sistema i predviđenog načina funkcionisanja aplikacije potrebno je definisati klase, ili kako je to u *Solidity*-u nazvano ugovore, čvor i korisnik.

Atribute ugovora *Node* (engl. *node*, srp. čvor) čine adresa naloga koji je ulogovan na uređaju i pokrenut *Geth* funkcijom, te radi lakše manipulacije objektom vrata dodjeljuje mu se indeks broj. Za atribute ugovora potrebno je realizovati *set* i *get* funkcije.

Ugovor *User* (engl. *user*, srp. korisnik) koristi atribute koji opisuju osobe koje se dodaju u bazu podataka. U konstruktoru ugovora prosljeđuju se fraza, lozinka, ime i indeks novog korisnika dok se ostali atributi inicijalizuju na nulu. Za atribute potrebno je realizovati *set* i *get* funkcije.

Kako bi se koristili forimirani ugovori u jednom Pametnom ugovoru, potrebno je napisati glavni ugovor koji je nazvan *Admin*. U *Admin* ugovoru formira se niz objekata ugovora *User* i niz objekata ugovora *Node* kao i jedna dodatna string promjenjiva u kojoj će se čuvati IPFS heš klasifikatora koji se dobija kao rezultat obuke neuronske mreže za prepoznavanje lica. Sa obzirom da će se na mrežu postaviti samo *Admin* ugovor, jedino će se moći pristupiti funkcijama i promjenjivim koje su u njemu napisane, naravno ukoliko su inicijalizovane kao javne. Konstruktor *Admin* ugovora je ralizovan tako da se prilikom postavljanja pametnog ugovora na mrežu čuva adresa *Ethereum* naloga inicijatora transakcije u promjenjivu admin. Suštinski, *set* i *get* funkcije napisane u ovom ugovoru pozivaju *set* i *get* funkcije napisane u ugovorima *User* i *Node*, sa tim da se, prilikom poziva funkcije, navodi neki parametar, recimo indeks ili korisničko ime, na osnovu kojeg se iz niza objekata odabira traženi korisnik ili čvor čiji se podaci mijenjaju, odnosno preuzimaju. Takođe realizovani su modifikatori koji predstavljaju specijalne funkcije koji dozvoljavaju ili ne dozvoljavaju transakciju, u zavisnosti od toga da li je zadovoljen specijalni uslov koji je napisan unutar samog modifikatora.

Specijalno napisane funkcije koje se pozivaju putem korisničke sprege se odnose na dodavanje i brisanje korisnika, dodavanje čvorova te provjeru i promjenu pristupa korisnika:

- *addUser* - Funkcija poziva konstruktor ugovora *User* i dodaje novog korisnika kao objekat u nizu korisnika.
- *addNode* - Funkcija poziva konstruktor ugovora *Node* i dodaje čvor kao objekat u nizu čvorova.
- *removeUser* - Kao ulazni parametar prosljeđuje se indeks broj korisnika koji će biti obrisan iz baze podataka, nakon čega se indeksi ostalih korisnika preraspoređuju.
- *changeUserAccess* - Kao ulazne parametre funkcija prima indeks broj korisnika kojem se vrši izmjena mogućnosti pristupa, indeks broj vrata na koja se mogućnost pristupa odnosi kao i vremena početka i kraja mogućnosti pristupa.
- *checkUserAccess* - Kao ulazne parametre funkcija prima korisničko ime i izrečenu govornu frazu a kao izlazni parametar daje bool vrijednost koja označava da li osoba ima mogućnost pristupa.

3.3. Realizacija koda za interakciju sa IPFS-om

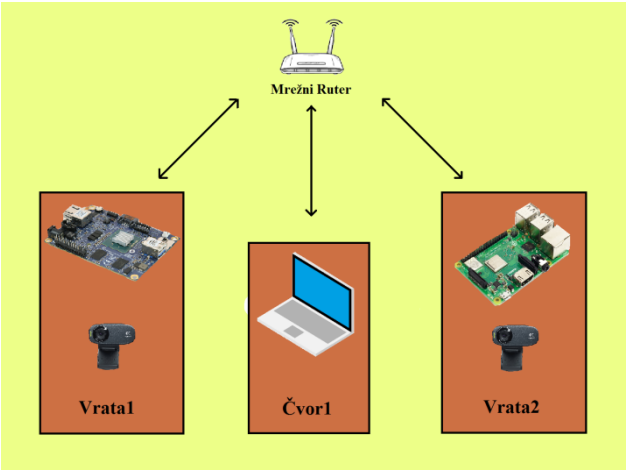
Iako je korisnička sprega napisana u *JavaScript* programskom jeziku, usljed toga što IPFS zahtjeva pristup podacima koji se nalaze na disku računara, ovaj dio koda je napisan u *Python* programskom jeziku. Za pisanje koda u *Python*-u, potrebno je instalirati neophodne biblioteke:

- *web3.py* [7] - je biblioteka za interakciju sa *Ethereum* blokčejnom.
- *ipfshttpclient.py* [8] - je bibilioteka za interakciju sa IPFS mrežom.

Sa obzirom da će se ovaj dio koda pozivati iz *JavaScript* aplikacije, prilikom određenih događaja u sistemu, potrebno je omogućiti proslijeđivanje parametara kao što su indeks broj korisnika. Ovo je realizovano kreiranjem JSON datoteke u kojoj se proslijeđuju indeks korisnika i programski režim.

4. DECENTRALIZOVANA APLIKACIJA

4.1. Princip funkcionisanja aplikacije



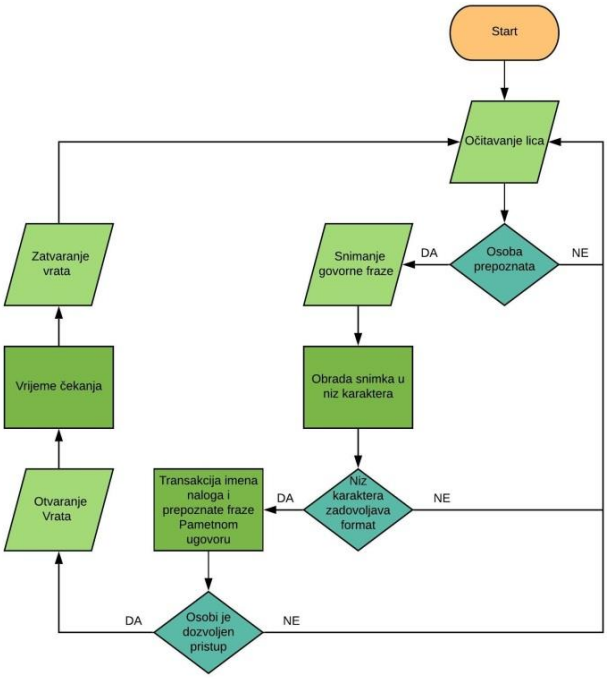
Slika 2: *Povezivanje fizičkih komponenti sistema*

Prije pokretanja decentralizovane aplikacije potrebno je ostvariti povezivanje fizičkih komponenti sistema prema slici 2. Prva vrata su upravljana korištenjem *Minnowboard Turbot Quad* računara, dok su druga vrata upravljana korištenjem *Rasppbery Pi 3 B+* računara. Na svim vratima se zahtjeva postavljanje kamere sa mikrofonom. Čvor 1 predstavlja računar koji mora imati značajno bolje karakteristike fizičke arhitekture, u poređenju sa *Minnowboard* i *Raspberry Pi* računarima, kako bi se što brže izvršila obuka mašinske vizije.

Nakon povezivanja komponenti fizičke arhitekture, potrebno je povezati sve komponente programske podrške u jednu funkiconalnu cjelinu. Programska podrška je podijeljena na tri distinktivna dijela:

1. Programska podrška za prepoznavanje lica je zasnovana na biblioteci *TensorFlow* [9] sistema *FaceNet* [10] za prepoznavanje lica..

2. Programska podrška za prepoznavanje govorne fraze je zasnovana na korišćenju *Python* biblioteke *SpeechRecognition* [11] koja razvijena za potrebe aplikacija prepoznavanja govora
3. Programska podrška za čuvanje i prikaz podataka zasnovana na korištenju blokčejn i IPFS tehnologija .



Slika 3: *Algoritam za utvrđivanje mogućnosti pristupa*

4.2. Analiza performansi sistema

Nakon pokretanja aplikacije, vrši se analiza performansi sistema. Na računarskim uređajima koji upravljaju radom vrata, u početku se pokreću potpuni *Ethereum* čvorovi, nakon čije analize performansi se pokreću laki *Ethereum* čvorovi.

Nakon testiranja, dobijeni su sledeći rezultati:

- Pokretanje potpunog Ethereum čvora je uspjelo na *Raspberry Pi 3 B+* uređaju, međutim ubrzo, nakon manje od deset potvrđenih blokova, gubi sinhronizaciju sa mrežom. Kada je na uređaj uspješno pokrenut laki čvor, uređaj je zadovoljio predviđene funkcionalnosti prema algoritmu prikazanom na slici 3. Ono što je bitno naglasiti jeste da usljed slabih karakteristika fizičke arhitekture, rad aplikacije je značajno spor, sa prosječnim vremenom od očitavanja lica osobe do otvaranja vrata 12s.
- Nakon pokretanja potpunog Ethereum čvora na *Minnowboard Turbot Quad* uređaju nije došlo do gubitka sinhronizacije sa mrežom. Aplikacija je funkcionisala bez problema, sa prosječnim vremenom od očitavanja lica osobe do otvaranja vrata od 7s. Nakon pokretanja lakog čvora, pokazalo se da je potrebno u prosjeku 6s od očitavanja lica do otvaranja vrata.

5. ZAKLJUČAK

U ovom radu je realizovan sigurnosni sistem zadužen za kontrolu pristupa osoba određenim oblastima. Sistem je zasnovan na korištenju biometrijskih karakteristika lica i karakteristične fraze na osnovu kojih se utvrđuje da li osoba ima dozvoljen pristup, poređenjem sa podacima u bazi podataka. Ova baza podataka je razvijena korištenjem *Ethereum* blokčejn tehnologije i naziva se pametni ugovor. Pametni ugovor sadrži sva pravila zapisana u linijama koda na osnovu kojeg funkcioniše decentralizovana aplikacija.

Nakon analize performansi sistema, pokazalo se da uređaj *Minnowboard Turbot Quad* zadovoljava predviđene funkcionalnosti pametnih vrata, dok uređaj *Raspberry Pi 3 B+* djelimično zadovoljava. Pored lošijih performasni aplikacije pokrenutoj na *Raspberry* uređaju, objašnjenje jeste nepouzdanost lakog čvora s obzirom da je još u fazi razvoja, pored toga laki čvor ne podržava mogućnost rudarenja.

Kao ciljevi nekog daljeg istraživanja može se zadati razvoj blokčejn mreža sa *Proof Of Authority* [4] konsenzus modelom. Na ovaj način se može dati pravo potvrđivanja transakcija nalozima koji su ulogovani na uređajima zaduženim za upravljanje vrata, što eliminiše potrebu za postojanjem rudara. Kao alternativa blokčejn tehnologiji, može se razmotriti razvoj programske podrške za skladištenje podataka zasnovanom na korištenju drugih DLT tehnologija koje zahtjevaju manje računarskih resursa.

6. LITERATURA

[1] https://en.wikipedia.org/wiki/Automated_fingerprint_identification (pristupljeno u oktobru 2019.)
[2] https://www.aware.com/biometric-identification-system-abis/ (pristupljeno u oktobru 2019.)
[3] Jitendra Chittoda, *"Mastering Solidity".*
[4] Debajani Mohanty, *"Ethereum For Architects and Developers, With Case Studies and Code Samples in Solidity"*.
[5] https://ethereum.org/ (pristupljeno u oktobru 2019.)
[6] https://ipfs.io/ (pristupljeno u oktobru 2019.)
[7] https://web3py.readthedocs.io/en/stable/ (pristupljeno u novembru 2019.)
[8] https://pypi.org/project/ipfshttpclient/ (pristupljeno u novembru 2019.)
[9] https://www.tensorflow.org/ (pristupljeno u novembru 2019.)
[10] Florian Schroff, Dmitry Kalenichenko, James Philbin, "*FaceNet: A Unified Embedding for Face Recognition and Clustering.pdf*"
[11] https://pypi.org/project/SpeechRecognition/ (pristupljeno u novembru 2019.)

Kratka biografija:

Živko Mišić rođen je u Banja Luci, BiH 1995. god. Osnovne studije završio na Fakultetu tehničkih nauka, smjer Mehatronika 2018.god.

Kontakt: zivan95misic@gmail.com