Primjer baze podataka u sustavu za upravljanje bazama podataka Neo4j

Pračić, Zlatko

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Zlatko Pračić

PRIMJER BAZE PODATAKA U SUSTAVU ZA UPRAVLJANJE BAZAMA PODATAKA NEO4J

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Zlatko Pračić

Matični broj: 0016145097

Studij: ***Primjena informacijske tehnologije u poslovanju***

PRIMJER BAZE PODATAKA U SUSTAVU ZA UPRAVLJANJE BAZAMA PODATAKA NEO4J

ZAVRŠNI RAD

Mentor:

Izv. prof. dr. sc. Markus Schatten

Varaždin, rujan 2022.

Zlatko Pračić

Izjava o izvornosti

Izjavljujem kako je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

U ovom radu bit će prikazan način razvoja aplikacije koja temeljem unesenih događaja kreira rang listu moguće zlostavljane djece na čijem je vrhu najugroženije dijete. Aplikacija se pokreće u web pregledniku i omogućuje unos događaja, ovisno o instituciji u kojoj se nalazi korisnik aplikacije te pregled liste djece za koju postoji sumnja da su ugroženi zlostavljanjem. Aplikacija pruža i vizualni pregled slučaja koji se kreira nakon unosa OIB-a osobe. Aplikacija pruža mogućnost autentifikacije i autorizacije i njenim mogućnostima se pristupa tek nakon potvrđivanja svog identiteta. Da bi se aplikacija sačinila korišten je Node.js sa svojim dodacima dostupnim preko npm (*Node Packet Manager*), a u pozadini se nalazi polustrukturirana baza podataka smještena na Neo4j serveru. U prvom dijelu rada se kratko opisuje fenomen zlostavljanog djeteta nakon čega se daju početne teoretske osnove grafičkih baza podataka. Nakon navedenog se opisuje način izgradnje aplikacije te potom korištenje iste kroz pogled jednog od korisnika.

Ključne riječi: zlostavljanje djeteta, baza podataka, polustrukturirana baza podataka, Neo4j, Node.js.

Sadržaj

1. Uvod

Tema završnog rada je aplikacija koja se temelji na sustavu upravljanja bazama podataka Neo4j. Sustav Neo4j pruža mnoge mogućnosti, a isti najviše dolazi do izražaja kada imamo podatke koji nisu do kraja strukturirani te je naročito koristan alat pri pretraživanju veza između pojedinih elemenata. Na svom web mjestu Neo4j Inc. spominje najvažnije načine primjene njihova sustava kao što su: prepoznavanje prevare, upravljanje znanjem, upravljanje identitetima i pristupom resursima, mrežne i IT operacije. U ovom radu se Neo4j koristi kako bi se mogla stvoriti slika o eventualnom zlostavljanju djeteta. Nakon popunjavanja pojedinačnih podataka od strane djelatnika različitih institucija i upita koji uzima u obzir sve te elemente stvara se rang lista moguće zlostavljane djece.

Tijekom izrade ovog rada korištena je relevantna literatura i znanstveni radovi koji opisuju rad s Neo4j bazama podataka. Isto tako su se rabili internetski članci o Neo4j i mrežno mjesto kompanije Neo4j Inc. Pored toga, za potrebe modeliranja aplikacije uporabljeni su različiti članci i znanstveni radovi koji govore o zlostavljanju djece.

Pri izradi aplikacije koristio se sustav Neo4j, konkretno Neo4j AuraDB (grafička baza podataka kao servis) koji u svojoj besplatnoj verziji nudi mjesto na online serveru s određenim ograničenjima u količini čvorova (200.000 čvorova) i relacija (400.000 relacija). Aplikacija je pisana uz pomoć integriranog razvojnog okruženja Visual Studio Code pri čemu se koristio Node.js s određenim dodacima koji će biti objašnjeni u samom radu. Pri kreiranju podataka rabljen je programski jezik Python kako bi se odradio web *scraping* (prikupljanje podataka s weba) te su poslije toga ti podaci pretočeni u csv (eng. – *comma-separated values* – vrijednosti odvojene zarezima) datoteku zbog lakšeg i preglednijeg rukovanja s istim.

Ovaj rad se dijeli na osam poglavlja od čega su prvo poglavlje uvod i zadnje, osmo poglavlje, zaključak. U drugom poglavlju se govori o fenomenu zlostavljanja djeteta, a pošto taj pojam predstavlja centralni dio aplikacije bilo je potrebno isto objasniti i izvući najvažnije pojmove vezane uz isto. U trećem poglavlju se govori o grafičkim bazama podataka, povijesti nastanka i specifičnostima. U četvrtom poglavlju se pobliže upoznaje s Neo4j sustavom za upravljanje bazama podataka. Peto poglavlje opisuje kreiranje grafičke baze podataka i popunjavanje iste s podacima potrebnim za rad. U šestom poglavlju je opisan način izrade aplikacije, a sedmo poglavlje daje prikaz korištenja aplikacije od strane korisnika.

Odlukom o odabiru sustava koji će se obraditi ovim završnim radom nije se ni slutilo koje su njegove mogućnosti. Nakon kratkog proučavanja Neo4j sustava za upravljanje grafičkim bazama podataka i upoznavanja s mogućnostima te razgovora autora završnog rada sa suprugom koja je netom diplomirala na Sveučilištu Sjever s temom „Kvaliteta života

zanemarene i zlostavljane djece i njihova integracija u društvo“ došlo se do zaključka kako bi se mogla napraviti aplikacija koja će upozoravati na moguće zlostavljanje djece. Ideja je bila da aplikacija sinergijski objedinjuje pojedinačno djelovanje djelatnika u različitim institucijama koje imaju dodira s fenomenom zlostavljanja djece. Taj pojedinačni rad raznih djelatnika bi nakon nekog vremena proizveo veliku, zajedničku sliku koja može upozoravati na moguće zlostavljanje djeteta. Naravno, aplikacija **ne daje podatak koje dijete jeste zlostavljano**, niti se na krajnji proizvod treba gledati na takav način. Aplikacija samo **daje upozorenje kako zlostavljanje možda postoji** zbog velikog broja povezanih i nepovezanih situacija u dječjem okruženju. I dalje ostaje na ljudima i institucijama odgovornost da zlostavljanje djeteta svojim operativnim radom prepoznaju i preveniraju.

Na žalost, često se u novinama može pročitati kako je zbog slabe suradnje institucija došlo do tragedije u kojoj su stradala djeca. Svi su svjesni kako upravo manjak suradnje dovodi do tragedije, ali o tome se priča par dana i poslije toga sve pada u zaborav do sljedećeg slučaja. Aplikacija izrađena za potrebe ovog rada svakako nije konačni odgovor na taj problem, ali pokazuje put kojim bi se moglo krenuti kako bi se postiglo sigurnije društvo, ne samo za djecu, već i za sve članove društva. Na samom početku izrade aplikacije autor je kontaktirao dvije institucije u nadi da će dobiti određene statističke pokazatelje i korelacije među pojedinim situacijama, ali na žalost dobio se hladan birokratski odgovor. Unatoč takvom odgovoru, autor je nastavio s potragom za literaturom koja govori o tom problemu i ipak se došlo do podataka, većinom u stranoj literaturi, vezanih uz temu koji će se kasnije iskoristiti u samom modeliranju aplikacije.

2. Fenomen zlostavljanja djeteta

Prije nego što se krene u razradu teme, ali i prije nego se krenulo u izradu aplikacije, bilo je potrebno razjasniti što je to zlostavljanje djeteta i kako isto prepoznati. Bilo je važno prepoznati simptome koji upućuju na to da je dijete zlostavljano kako bi isti bili iskorišteni kao dio podataka koji prezentiraju taj dio realnog svijeta [1].

2.1. Osnovni pojmovi

Kako bi se ispravno odredile vrijednosti koje određuju što je to čvor (eng. *node*) i odnos (eng. *relationship*) te koja su njihova svojstva (eng. *properties*) potrebno je definirati pojedine pojmove. Konvencija o pravima djeteta definira pojam djeteta pa tako osoba mlađa od 18 godina je dijete [2].

Po Svjetskoj zdravstvenoj organizaciji maltretiranje djece je zlostavljanje i zanemarivanje koje se javlja kod djece mlađe od 18 godina. Uključuje sve vrste fizičkog i/ili emocionalnog zlostavljanja, seksualnog zlostavljanja, zanemarivanja, nemara i komercijalnog ili drugog iskorištavanja, koje rezultira stvarnom ili potencijalnom štetom po zdravlje, opstanak, razvoj ili dostojanstvo djeteta. Nasilje prema partneru također se ponekad uključuje kao oblik maltretiranja djece [3].

2.2. Indikatori zlostavljanja

Struktura obitelji, osobito kada je ista nestabilna, povezana je s povećanim rizikom od zlostavljanja djece. **Samohrano roditeljstvo** više nego udvostručuje rizik od uključenosti službe za zaštitu djece, a jedna studija je pokazala da je to drugi najveći čimbenik rizika zlostavljanja djece, nakon prihoda. Samohrano roditeljstvo doprinosi financijskom stresu, društvenoj izolaciji i nedostatku društvene podrške, a sve to povećava vjerojatnost zlostavljanja djece. Slično, status očinstva povezan je s rizikom maltretiranja djece, a prisutnost očinskih figura koji nisu biološki očevi, osobito kada su višestruki i prolazni, povećava rizik od seksualnog zlostavljanja. Drugi značajni obiteljski čimbenici rizika zlostavljanja djece su obiteljski sukobi te nezadovoljstvo u braku. Nadalje, siromaštvo, nezaposlenost i broj uzdržavanih osoba u kućanstvu dodatno opterećuju obitelji i vode prema ovisnosti o **socijalnoj pomoći**, a svi ti rizici povećavaju rizik od zlostavljanja djece. Veći broj djece u obitelji povećava vjerojatnost zanemarivanja djece i vjerojatnost dodira obitelji sa službom za zaštitu djece. Nestabilne obiteljske strukture i problematične obiteljske situacije predviđaju zlostavljanje djece i obiteljsko nasilje. Izloženost nasilju u obitelji povećava rizik zlostavljanja i zanemarivanja [4].

Društvena izolacija te nedostatak financijskih i materijalnih sredstava vode ka fizičkom zlostavljanju djece. Majke s lošijom društvenom povezanošću suočavaju se s dvostruko većim rizikom od maltretiranja svoje djece. Niski prihodi i siromaštvo povezani su s fizičkim zlostavljanjem i zanemarivanjem djece [4].

Roditeljska **zlouporaba droga** više nego udvostručuje rizik da dijete bude izloženo fizičkom ili seksualnom zlostavljanju. Utvrđeno je da roditeljska zlouporaba droga povećava manje nasilna djela roditelja za 20%, a nasilnija za 46% [4].

Ako je roditelj bio zlostavljan u djetinjstvu isto značajno povećava rizik da će taj roditelj zlostavljati svoje dijete. Karakteristike majke, uključujući dob i obrazovanje utiču na zlostavljanje djeteta. Mladi roditelji imaju tri puta veću vjerojatnost da će maltretirati svoju djecu, a roditelji s niskim obrazovnim stupnjem su u pet puta većem riziku. Mlada dob majke i nisko obrazovanje predviđaju uključenost službe za zaštitu djece, povećavajući i vjerojatnost istrage te upisa djeteta u registar za zaštitu djece [4].

Često se osoblje u raznim institucijama pita kada treba djelovati? Mora se djelovati čim postoji sumnja u zlostavljanje. Pri tome organizacije moraju zauzeti stav o nekažnjavanju prijava koje će se kasnije eventualno pokazati neutemeljenim. Ukoliko bi se kažnjavala dojava koja bi na kraju ipak dovela do toga da je neutemeljena, osoblje bi se ustručavalo od prijavljivanja i samim time bi većina zlostavljanja prošla neregistrirano. Uobičajeni fizički pokazatelji zlostavljanja djece su **modrice**, udubljenja, posjekotine/ogrebotine ili opekline (posebno one na leđima, nogama, rukama i unutarnjoj strani bedara ili na neuobičajenim mjestima i mogu nalikovati predmetu), unutarnje ozljede i **prijelomi kostiju** koji **nisu u skladu s ponuđenim objašnjenjem,** bilo kakve ozljede genitalnog ili rektalnog područja (npr. modrice, krvarenje, infekcija ili nešto što uzrokuje bol pri odlasku na zahod). Nadalje, nošenje odjeće neprikladne vremenskim uvjetima kako bi se sakrile ozljede, spolno prenosive bolesti i/ili česte infekcije mokraćnih puteva su isto tako pokazatelji zlostavljanja djeteta. Isti tako djeca koja izgledaju stalno prljava i neoprana i/ili neprikladno odjevena (prema vremenskim uvjetima), stalno gladna, umorna i bezvoljna, koja često imaju zdravstveni problemi i kod kojih se primijeti nedostatak rutinske medicinske njege ozljeda su vjerojatno zlostavljana ili zanemarena [5].

Uobičajeni ponašajni pokazatelji zlostavljanja djece su njihovi crteži ili pisanje koji prikazuje nasilje i zlostavljanje te **redoviti izostanak iz škole bez razumnog objašnjenja**, značajna i neobjašnjiva kašnjenja u emocionalnom, mentalnom ili fizičkom razvoju. Regresivne ili neobične promjene ponašanja (npr. iznenadni pad u napretku u školi, nervoza, depresija, **povlačenje**, hiperaktivnost, agresija, mokrenje u krevet), zlouporaba droga ili alkohola, samoubojstvo ili samoozljeđivanje, nanošenje štete drugima ili životinjama, nedosljedno ili

malo vjerojatno objašnjenje ozljede ili nemogućnost pamćenja uzroka su također neki od pokazatelja zlostavljanja. Nevoljkost da se ide kući i/ili oprez te strah od roditelja ili skrbnika, neobičan strah od fizičkog kontakta s odraslim osobama, dobno neprikladna seksualna aktivnost (npr. pretjerana masturbacija ili trljanje genitalija o odrasle osobe, promiskuitet), loša njega ili osobna higijena ukazuju na moguće zlostavljanje. Neobično bliska povezanost sa starijom osobom koja poklanja skupe darove ili novac (npr. novi mobitel koji im je dao "prijatelj"), prerano preuzimanje uloge skrbnika pri čemu pokušava zaštititi druge članove obitelji je također jedan od pokazatelja zlostavljanja djeteta [5].

Uobičajeni pokazatelji zlostavljanja djece od strane odraslih članova obitelji (roditelji, braća i sestre, šira obitelj) su pokušaji jednog roditelja da otuđi svoje dijete od drugog roditelja, pretjerano zaštitnički ili nestabilni odnosi, nevoljkost djeteta da bude samo s jednim ili više članova svoje obitelji. Brat ili sestra koji se ponaša kao dečko i djevojka (sramota ako se nađu sami zajedno) je isto tako jedan od pokazatelja. Druge odrasle osobe (npr. član školskog osoblja, treneri) koji neprikladno dodiruju dijete, iznose seksualni materijal, neprikladan kontakt s djetetom (npr. pozivi, e-poruke, poruke, društvene mreže), očiti ili neprikladni preferencijalni tretmani dijete (zbog čega se osjeća "posebno"), davanje neprikladnih i skupih darova djetetu, neprikladne društvene granice (npr. pričanje djetetu o svojim osobnim problemima), nuđenje da odveze dijete u školu ili iz škole, pozivanje djeteta u svoj dom su također jedan od mogućih pokazatelja [5].

Zlouporaba droga ima veliki utjecaj na sustav skrbi za djecu. Procjenjuje se da 9 posto djece u Sjedinjenim Američkim Državama (6 milijuna) živi s barem jednim roditeljem koji zlorabi alkohol ili droge. Istraživanja su pokazala da je vjerojatnije da će djeca roditelja koji su ovisnici o drogama doživjeti zlostavljanje ili zanemarivanje nego djeca u kućanstvima gdje se ne zlouporabljuju opojna sredstva [6].

Pojedini najvažniji indikatori koji su prepoznati kao elementi koji upućuju na moguće zlostavljanje djeteta su u ovom odjeljku podebljani i isti se koriste u aplikaciji u obliku svojstva čvora kojim se bilježi određeno ponašanje ili događaj.

2.3. Rad institucija u RH

Jedan od primjera gdje je rezultat zlostavljanja i zanemarivanja djeteta smrtno stradavanje jedne mlade osobe (16) primjer je iz Malog Lošinja. Isti nastaje uslijed eskalacije nasilja kojeg je počinio žrtvin otac. Počinitelj je otprije bio poznat Policiji zbog nasilja u obitelji te posjedovanja droge, a slučaj se desio 2020 [7].

Ovaj slučaj je spomenula Pravobraniteljica za djecu u svom izvješću i obilježen je kao eklatantan primjer nedostatka suradnja između nadležnih institucija. U izvješću se upozorava

kako 2014. nije prihvaćen njihov prijedlog koji bi poboljšao položaj djeteta u odnosu na osobe s duševnim teškoćama. I dalje se zbog obveze čuvanja privatnosti pacijenta ne dostavljaju podaci ostalim službama kada bi došlo u pitanje zaštita interesa djeteta. Čak i u situacijama kada se radi o osobama koji su registrirani u centrima socijalne skrbi kao korisnici droge ili su počinitelji nasilnog ponašanja zdravstveni sustav ne dostavlja podatke ostalim zainteresiranim stranama koje bi trebale raditi u interesu zaštite prava djeteta. Zbog svega navedenog pravobraniteljica poziva i upozorava kako je prijeko potrebno poboljšati suradnju i koordinaciju između resora koji među svojim ingerencijama imaju pitanje zaštite djeteta. To posebno dolazi do izražaja ako se radi o osobama koje imaju povijest nasilnog ponašanja. Izvješća pravobraniteljice za djecu koji se odnose na 2019., 2020. i 2021. kažu kako je u te tri godine bilo ugroženo pravo na život za 15 djece u 12 predmeta. Pored toga bilo je ugroženo pravo na zaštitu od nasilja za 907 djece u 581 predmeta [8] [9] [10].

3. Grafičke baze podataka

Danas su inače više prihvaćeni sustavi za upravljanje bazama podataka koji se temelje na relacijskima bazama podataka i isti rade sa skupovima podataka koji su međusobno povezani relacijama, a za rad s tim podacima se koristi SQL (eng. *Structured Query Language*) upitni jezik. Takvi sustavi koriste tablice s podacima koje su međusobno povezane. Ali pored takvih sustava sve su popularnije i NoSQL baze podataka [11]. Premda bi mnogi pretpostavili kako NoSQL znači “ne SQL-u”, zapravo je pravi smisao akronima "ne samo SQL" i isti su alternativa relacijskim bazama podataka [12].

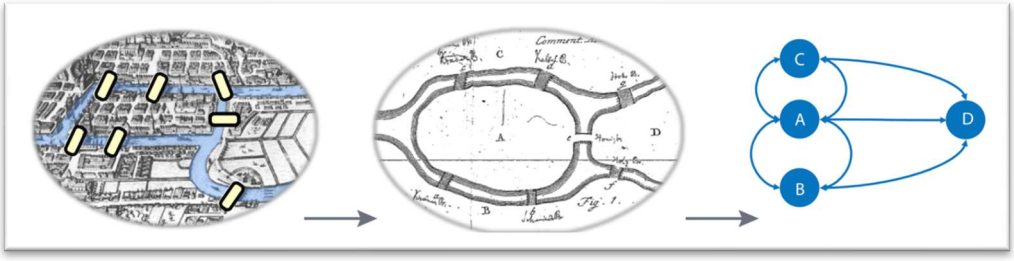
Neo4j je jedan od predstavnika NoSQL sustava za upravljanje bazama podataka i izuzetno je popularan zbog velike brzine čitanja i pisanja. Fleksibilan je pri dizajnu podataka, izuzetno je prilagodljiv, jednostavan je za korištenje i praktičan za modeliranje. Visokih je performansi i pohranjuje strukturirane podatke u mreži (grafovi) umjesto u tablice [13].

Kako bi se nekome približio sustav grafičkih baza podataka prvo je potrebno objasniti pozadinu tog sustava, a najbolje je krenuti sa samom teorijom grafova i pojmom polustrukturiranih baza podataka.

3.1. Teorija grafova

Povijest teorije grafova počinje s Leonhardom Eulerom, švicarskim matematičarem i fizičarem. Euler je dao mnogo značajnih doprinosa čistoj i primijenjenoj matematici tijekom više od 50-godišnje akademske karijere. Njegovo rješenje problema sedam Königsberških mostova iz 1735. smatra se prvim teoremom teorije grafova i jednim od njegovih najvažnijih doprinosa znanosti. Problem sedam Königsberških mostova je bio pronaći put kroz grad koji će prijeći svaki od sedam mostova koji povezuju dva velika otoka [14].

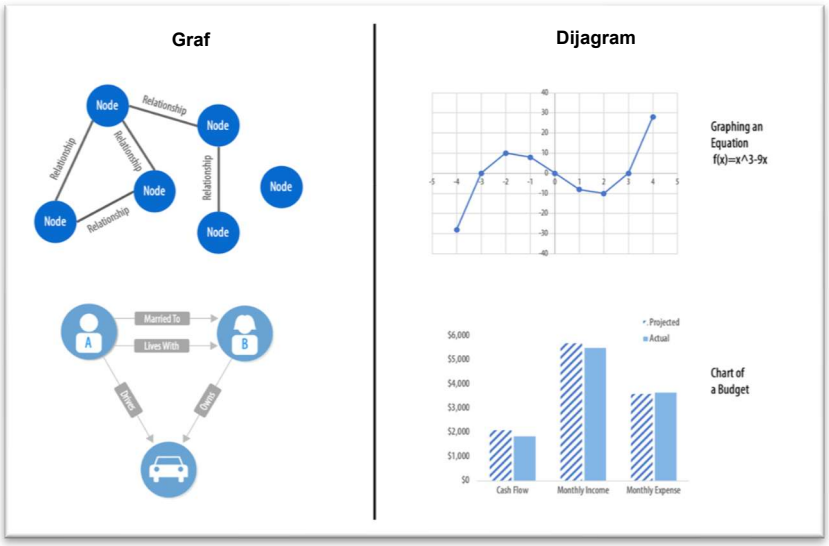
Drugi uvjeti problema bili su da se mostovi ne smiju prijeći više od jednom i da se svaki most mora prijeći u potpunosti. Eulerova kasnija rasprava o problemu napisana je 1736. i objavljena 1741. Euler je dokazao kako se problem ne može riješiti, ali što je još važnije, primijetio je da je najrelevantniji aspekt problema redoslijed kojim su se mostovi prešli. U stvaranju ovog jedinstvenog, temeljnog pristupa, Euler je mogao ispitati problem u apstraktnim terminima. Njegova metodologija razmatrala je samo kopno, otoke i mostove koji su ih povezivali [14].



Slika 1 – Počeci teorije grafova [14]

U teoriji grafova, kopno i otoci su ono što se naziva vrhovima. Svaki most koji povezuje dva vrha poznat je kao brid, koji, za potrebe teorije grafova, služi za identifikaciju koji je par vrhova povezan tim mostom. Na Slici 1 mogu se vidjeti komponente problema raščlanjene na četiri vrha povezana sa sedam bridova. Konačna matematička struktura koja predstavlja sve vrhove i bridove naziva se graf [14].

Često se znaju miješati pojmovi grafa (skraćeno od mrežni graf) i dijagrama (Slika 2). Iako podaci ili dijagrami grafikona mogu biti sadržani u grafikonu, pojmovi graf i dijagram nisu sinonimi [14]. U skladu s Hrvatskom enciklopedijom „mrežni graf je matematički objekt u teoriji grafova koji se sastoji od konačnog skupa vrhova i konačnog skupa bridova (lukova, grana, dužina koje povezuju vrhove). Vrhovi se prikazuju kao kružići, a bridovi kao linije koje spajaju vrhove“ [15]. S druge strane, u skladu s istom enciklopedijom „dijagram je grafički prikaz vrijednosti veličina određenih pojava i odnosa u prirodi i društvu, predočen geometrijskim tijelima i likovima“ [16].



Slika 2 – Prikaz razlike između grafa i dijagrama [14]

3.2. Polustrukturirane baze podataka

Uslijed brzog rasta World Wide Web-a i elektroničkog poslovanja došlo je do potrebe za razmjenom različitih podataka. Kako su svi ti podaci bili različitog oblika i sadržaja te nisu uvijek bili potpuni, razvijen je sustav koji bi rukovao s takvim, nepotpunim to jest polustrukturiranim podacima. Upravo je najveća prednost takvog sustava mogućnost prezentiranja podataka koji se ne mogu smjestiti u unaprijed određene strukture. Takvim sustavima je izuzetno lako dodavati nova svojstva ili preoblikovati format, a to je bilo potrebno web servisima koji su bili u razvoju i rastu. S druge strane, upravo ta njihova prednost jeste i najveći nedostatak pošto se zbog strukture podataka teško postavljaju upiti u čemu su strukturirane baze podataka puno efikasnije [17].

Tradicionalne relacijske baze podataka kod kojih se podrazumijeva kako su podaci strukturirani nisu prikladni web aplikacijama pošto podaci na kojima počivaju te aplikacije nisu strukturirani i moglo bi se reći kako su „nekompletni“. Tako mnoge tehnologije koje se koriste u relacijskim bazama podataka ne bi bile primjenjive. Ti slabije strukturirani podaci, poznatiji kao polustrukturirani podaci, često se predstavljaju kao stablo elemenata gdje je element „dijete“ pod-element „roditelja“. S druge strane, isti ti elementi mogu imati neka svoja svojstva. Kod njih su upiti nad stablima podataka predstavljeni kao izrazi putanje [18].

4. Neo4j sustav upravljanja bazama podataka

Pojavom NoSQL baza podataka, kao što su MongoDB i Cassandra predstavljen je jednostavniji način modeliranja podataka, ponuđen je visok stupanj fleksibilnosti i pristup oblikovanju bez obvezne sheme. U svom najjednostavnijem obliku, grafička baza podataka je skup vrhova i bridova. Jedan od načina prikaza grafičkih baza podataka je promatranje podataka kao proizvoljnog skupa objekata povezanih jednom ili više vrsta odnosa [12]. Sve su to koncepti koji se mogu pronaći u Neo4j sustavu za upravljanje bazama podataka (u daljnjem tekstu SUBP).

4.1. Povijest razvoja Neo4j

Neo4j je nastao 2000. kada su Emil Eifrem, Johan Svensson i Peter Naubauer (kreatori Neo4j) počeli primjećivati značajnu količinu troškova u izvedbi i radu u jednoj od njihovih aplikacija. Nakon što su tražili alternative i proveli nekoliko različitih istraživanja, započeli su s projektom „Project Neo“ koji je trebao dati odgovore na izazove s kojim su se susretali [12].

Projekt Neo je imao za cilj proizvesti SUBP koji nudi bolji način modeliranja, pohranjivanja i dohvaćanja podataka uz zadržavanje svih temeljnih koncepata koje su imale dotadašnji SUBP – poput ACID (eng. *Atomicity, Consistency, Isolation and Durability* to jest atomičnost, dosljednost, izoliranost i trajnost) te transakcije. To su koncepti koji moraju biti prisutni u jednom modernom SUBP [12].

Naknadna istraživanja i razvoj doveli su Neo4j među vodeća rješenja, opravdavajući slogan povezan s logotipom Neo4j na promotivnim materijalima „*The World's Leading Graph Database*“ [19].

4.2. Temeljni pojmovi

Kada se razgovara o grafičkim bazama podataka, vrhovi se najčešće nazivaju čvorovima, a rubovi odnosima ili relacijama [12].

Čvor se može smatrati objektom s bilo kojim brojem svojstava. Za razliku od ključeva koji povezuju retke unutar relacijske baze podataka, odnosi unutar grafičke baze podataka također mogu imati **svojstva**. Počevši od 2.0 verzije Neo4j, koncept oznaka uveden je kao način grupiranja čvorova. Čvor se može definirati kao „Osoba“, a zatim dati dodatne vrijednosti za svako svojstvo čvora prema potrebi. Grupiranjem čvorova na ovaj način, može se postaviti upit grafu da pokaže uobičajene pod-skupove onoga što su u biti tipovi čvorova. Označavanje

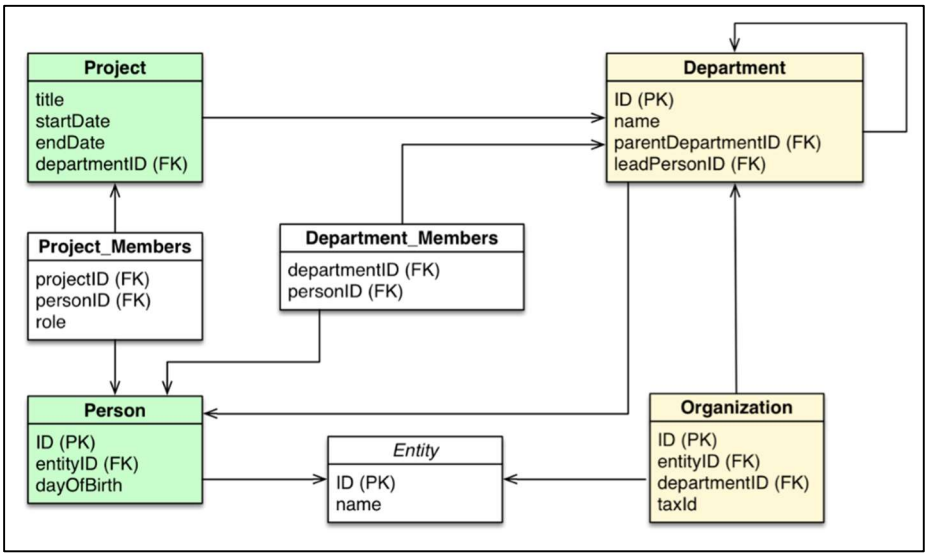
čvorova također nudi način za provođenje ograničenja modeliranja kada je to potrebno, kao i za povećanje brzine pristupa podacima kroz poboljšano indeksiranje [12].

Najčešća metoda za ispitivanje grafa je izvođenje **obilaska** (eng. *traversal*). U operaciji prelaska, upit počinje s jednim čvorom koji slijedi put odnosa preko povezanih čvorova. Kao i mnoge druge baze podataka, Neo4j se oslanja na **indeks** za eksplicitno traženje određenog čvora ili odnosa [12].

4.3. Relacijske baze podataka i Neo4j

Pri odabiru između grafičkih baza podataka i relacijskih baza podataka, jedna stvar koja bi trebala biti jasna na početku rada s jednim od ta dva sustava je oblik informacije. Grafičke baze podataka ili bilo koje druge NoSQL opcije ne mogu uvijek zamijeniti relacijske baze podataka. Međutim, ako postoje ograničenja definiranja unutar relacijske baze podataka, to bi bio razlog kad bi bilo preporučljivo prijeći na grafičku bazu podataka kao što je Neo4j. Drugi razlog zbog kojeg bi se moglo razmisliti o prelasku na grafičku bazu podataka je izbjegavanje polumjera i zaobilaznih rješenja koje morate koristiti kako bi se model uklopio u relacijsku bazu podataka [12].

Još jedan razlog za davanje prednosti grafičkim bazama podataka u odnosu na relacijske baze podataka jest izbjegavanje onoga što se naziva „*JOIN Hell*“ (pakao pridruživanja). Grafičke baze podataka takve probleme rješavaju puno elegantnije [12].



Slika 3 - Model primjera [20]

U članku „*RDBMS & Graphs: SQL vs. Cypher Query Languages*“ [20] naveden je primjer koji potvrđuje prethodnu izjavu i može se vidjeti u nastavku. Na Slici 3 je prikazan model

koji se koristi u primjeru. U organizacijskoj domeni prikazan na modelu potrebno je izvršiti upit koji navodi zaposlenike u "*IT Department*". Kada bi se koristio SQL jezik, upit bi izgledao na sljedeći način:

```
SELECT name FROM Person
LEFT JOIN Person_Department
  ON Person.Id = Person_Department.PersonId
LEFT JOIN Department
  ON Department.Id = Person_Department.DepartmentId
WHERE Department.name = "IT Department"
```

S druge strane, isti takav upit, ali uz uporabu Cypher (jezik kojeg koristi Neo4j, isti će biti obrađen u jednom od sljedećih potpoglavlja) bi izgledao na sljedeći način:

```
MATCH (p:Person)<-[:EMPLOYEE]-(d:Department)
WHERE d.name = "IT Department"
RETURN p.name
```

Iz navedenog je vidljivo koliko je jednostavnije postaviti upit putem Cypher, a osim toga, upit izgleda intuitivnije i lakši je za shvatiti.

Unatoč razlikama između grafičkih i relacijskih baza podataka, postoji nekoliko sličnosti. Najveća sličnost je da oba sustava postižu ono što je poznato kao usklađenost s ACID, što je skup principa koji jamče da se transakcije koje dovršava baza podataka obrađuje pouzdano [12].

4.4. NoSQL i Neo4j

Prva velika razlika između Neo4j grafičke baze podataka i drugih NoSQL kategorija je model podataka. Svaki tip čvora može imati bilo koji broj svojstava. Osim toga, ta se svojstva mogu mijenjati tijekom vremena, što daje model koji ne zahtijeva shemu. Kada se to spoji s činjenicom kako proizvoljni odnosi također mogu imati bilo koji broj vlastitih konfigurabilnih svojstava, razlika je još veća [12].

Pored toga, kako se grafikoni mogu brzo prilagoditi promjenama u poslovnim potrebama, posebno u povezivanju podataka, organizacijama je omogućeno da postavljaju upite kako se pojavi potreba pri tome ti upiti ne moraju biti precizno identificirani prije unosa podataka [12].

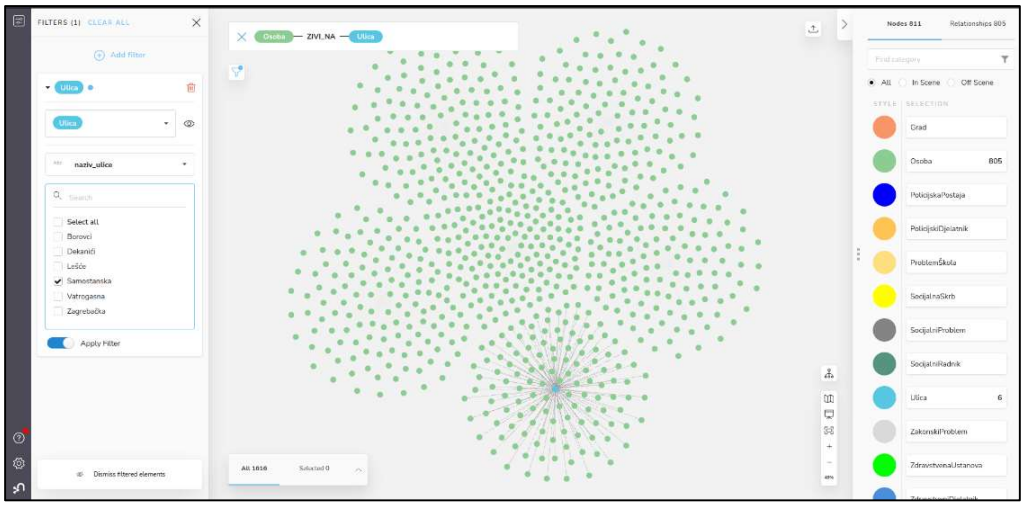
4.5. Instalacija

Neo4j je moguće instalirati na najveći broj dostupnih operacijskih sustava. Postoje verzije za Windows, Linux i MacOS operativne sustave. Instalacija je jednostavna, a nakon instalacije hardverski nije zahtjevna i omogućava ugodan daljnji rad [12].

Prije instalacije Neo4j je potrebno provjeriti jesu li softverski i hardverski preduvjeti na računalu ispunjeni. Što se tiče hardverskih zahtjeva, ako se Neo4j koristi za osobnu uporabu i razvoj minimalno je imati Intel Core i3 procesor, a preporučuje se i7, memorije je dovoljno za rad 2GB, a preporučuje se 16GB i na kraju potrebno je osigurati 10GB prostora na HDD. Na kraju samog imena SUBP Neo4j se nalazi slovo „j“ s razlogom. Isto označava Javu i Java Development Kit (JDK) koji je potreban za pokretanje Neo4j. Prije nego što se krene s instalacijom Neo4j potrebno je provjeriti je li na računalu instaliran Oracleov JDK. Ako je već instaliran JDK, potrebno je provjeriti koja je verzija instalirana. Ako se planira koristiti trenutno najnoviji Neo4j SUBP verzije 4.x tada na računalu treba biti instaliran JDK 11 i za Java Virtual Machine (JVM) Java SE 11 Platfor Specification. Ako se instalira Neo4j Desktop (verzija za rad na lokalnom računalu) s istim u kompletu dolazi i zadnja spomenuta komponenta (JVM) [21].

Neo4j se svakodnevno razvija. Kada se počela izrađivati aplikacija za potrebe ovog rada Neo4j je nudio Neo4j Desktop (za rad na lokalnom računalu) i Neo4j AuraDB (sustav u oblaku) uz koje su bili ponuđeni alati Neo4j Browser (za rad s bazom i grafički prikaz elemenata, Slika 4) te Neo4j Bloom (bazira se na grafičkom prikazu i pojednostavljuje upite na niz klikova mišem, Slika 5). Danas u ponudi Neo4j nudi i Neo4j Graph Dana Science i mrežni resurs Neo4j AuraDS [22].

Slika 4 – Uporaba Neo4j Browser na Aura DB platformi



Slika 5 – Uporaba Neo4j Bloom na Aura DB platformi

4.6. Cypher

Maloprije se spomenuo Cypher kad se radila usporedba sa SQL-om. Cypher je deklarativni jezik upita koji se koristi za rad s podacima u Neo4j. Po načinu rada sliči radu s relacijskim bazama podataka kroz strukturirani jezik upita (SQL) za izvođenje operacija s podacima. Zbog toga se kaže kako će onaj koji razumije SQL lako naučiti rabiti i Cypher [12].

Čvor se predstavlja okruglim zagradama, kao na primjer: (o:Osoba). Strelice odnosa su nacrtane na sljedeći način: -->, a dodavanje vrste relacije i druge informacije se nalaze u uglatim zagradama -[:OBITELJ]->. Spajanje ta dva elementa izgledaju kao dijagram kojeg je lako čitati i pratiti pa je vrlo lako shvatiti sljedeći dio upita: (o1:Osoba)-[r1:OBITELJ]->(o2:Osoba)-[p:ZAKONSKI]->(zp:ZakonskiProblem). Osoba o1 je u obiteljskoj relaciji s osobom o2 koja ima problem sa zakonom [23].

U nastavku će biti navedeni osnovni primjeri Cypher komandi uz pomoć kojih se upravlja Neo4j bazom podataka:

CREATE – poput INSERT u SQL-u. Služi kreiranju čvora/čvorova te relacije/relacija s i bez svojstava. Na primjer: CREATE (o:Osoba { ime : 'Zlatko', prezime: 'Pračić, datumRodjenja: '1972-02-25' }) stvara čvor Osoba koja ima sljedeća svojstva: ime – Zlatko, prezime – Pračić i datum rođenja – 25.02.1972. Isto tako može stvoriti i više čvorova pa na primjer CREATE (o:Osoba), (pp:PolicijskaPostaja) kreira čvor Osoba i PolicijskaPostaja. Moguće je i stvaranje relacija pa na primjer CREATE (o:Osoba)-[:ZAPOSLENIK]->(pp:PolicijskaPostaja) stvara dva čvora i međusobnu vezu između tih čvorova gdje je Osoba zaposlenik policijske postaje [24].

MATCH ima ulogu poput JOIN u SQL-u. MATCH omogućuje određivanja obrazaca koje će Neo4j pretražiti u bazi podataka. To bi bio glavni način dobivanja podataka u trenutnom skupu veza. Na primjer MATCH (a:Osoba {ime: 'Zlatko', prezime: 'Pračić'}), (b:Osoba {ime: 'Zlatko', prezime: 'Pavlić'}) CREATE (a) -[r:POZNAJE]-> (b) RETURN a, r, b traži osobe s datim svojstvima i stvara međusobnu vezu „POZNAJE“ između ta dva čvora [25].

I još jedna osnovna komanda je **SET**. Ista služi kako bi se ažurirala svojstva čvorova i relacija. Na primjer MATCH (n {ime: 'Andy'}) SET n.prezime = 'Taylor' RETURN n.ime, n.prezime dodaje svojstvo prezime čvoru n ili ako je bilo postavljeno neko drugo prezime, isti podatak mijenja i postavlja novi [26].

To bi bile osnovne komande, ali u nastavku rada će se spomenuti još neke koje su se rabile tijekom izrade aplikacije.

5. Popuna baze podacima

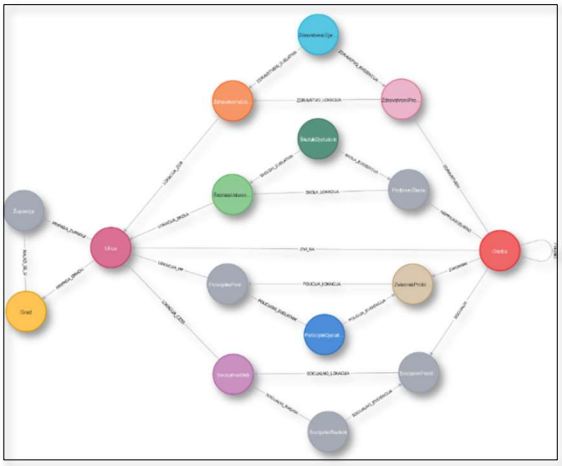
Kako je već u samom uvodu rečeno, aplikacija se radila uz pomoć Node.js radnog okvira. Ali, prije same izrade aplikacije bilo je potrebno izraditi model baze podataka te bazu napuniti bar početnim podacima. Neo4j podržava nekoliko načina popunjavanja podacima. Nakon popunjavanja temeljnim podacima pristupilo se izradi aplikacije u navedenom radnom okviru.

5.1. Modeliranje

Prije nego se krene s izradom aplikacije potrebno je izraditi bazu podataka koja će poduprijeti samu aplikaciju. Zbog toga je korisno sačiniti model podataka koji služi kao vizualni prikaz željenih podataka koji će se nalaziti u bazi i služiti kao podrška aplikaciji. Modeli predstavljaju objekte, kao što su korisnik ili institucija, veze između objekata i pravila koja određuju kako se objekti pohranjuju u bazi podataka. Model se obično koncentrira na to koji će podaci biti pohranjeni i kako će biti organizirani. Bez obzira na to koristi li se grafička baza podataka poput Neo4j ili relacijska baza podataka, modeliranje je ključan dio u osiguravanju da se podaci aplikacije mogu pohraniti i dohvatiti čim učinkovitije [12].

Shema u Neo4j se odnosi na indekse i ograničenja. Neo4j se često opisuje kao opcijska shema, što znači da nije potrebno stvarati indekse i ograničenja. Mogu se kreirati podaci - čvorovi, odnosi i svojstva - bez unaprijed definiranja sheme. Indeksi i ograničenja mogu se uvesti kada se želi, kako bi se postigle prednosti performansi ili modeliranja [27].

Pri izradi aplikacije za potrebe ovog rada sačinjen je model prikazan na Slici 6.



Slika 6 – Model rabljen u aplikaciji

Kako je autor želio popuniti bazu podataka podacima koji su lokalnog karaktera (domaća imena i prezimena te imena gradova, ulica, škola …) opcija uporabe Mockaroo (web servis za generiranje shema, setova podataka [28]) nije bila moguća. Navedeni servis se mogao koristiti samo pri generiranju OIB brojeva. Kako bi se došlo do imena koji su u uporabi u Republici Hrvatskoj primijenio se „web scraping“ tj. prikupljanje podataka s weba. Korištena je web stranica „Značenje imena“ s adresom https://www.znacenje-imena.com/ [29].

Inače, web scraping je izraz koji opisuje korištenje programa za preuzimanje i obradu sadržaja s weba. Za tu priliku je korišten programski jezik Python. Python je programski jezik (s pravilima sintakse za pisanje koda), ali i softver (Python interpreter) koji čita izvorni kod (napisan na jeziku Python) i izvodi njegove upute [30].

Program na kraju posprema imena u jednu datoteku s nastavkom csv. CSV datoteka je tekstualna datoteka koja ima određeni format koji omogućuje spremanje podataka u strukturiranom formatu tablice [31]. U nastavku se nalazi kod tog kratkog programa.

```
import requests
from bs4 import BeautifulSoup
import csv
from random import randint
from time import sleep

csv_datoteka = open('muska_imena.csv', 'w')
csv_writer = csv.writer(csv_datoteka)

for page in range(1, 19):
    url = "https://www.znacenje-imena.com/imena/hrvatska/muska?page="
    body = requests.get(url + str(page))

    body_text = body.content

    soup = BeautifulSoup(body_text, 'html.parser')

    for divs in soup.find_all('div', class_='col-sm-3 col-xs-6'):
        ime = divs.a.text
        brojevi = divs.span.text
        broj = brojevi[4:]
        csv_writer.writerow([ime, broj])

    sleep(randint(1, 2))

csv_datoteka.close()
```

Program u prvom dijelu učitava potrebne module nakon čega otvara csv datoteku. Potom ide na web mjesto, a pošto je to mjesto straničeno, na kraju web adrese se nalazi

varijabla koja prolazi vrijednosti od 1 do 19. Program traži klase elemenata određene programom i iz njih čita tekstualni podatak (ime) i brojčani podatak (broj sviđanja) koje posprema u csv datoteku. Da server na kojem se nalazi web stranica ne bi „pomislio“ kako je pod napadom, između prelazaka s jedne stranice na drugu prolazi slučajno generirano vrijeme. Inače, neki web serveri kod kojih je postavljena zaštita znaju blokirati IP adresu s koje stiže zahtjev za jako brzim pristupima stranicama. Svi ti podaci se pospremaju red po red u csv datoteku, a kad program prođe kroz sve svoje iteracije posprema datoteku. Slično se odrađuje i za ženska imena, samo je URL postavljen drugačije, kao i broj iteracija koji ovisi o straničenju.

Po prikupljanju muških i ženskih imena, te prezimena sa stranice https://www.mirovina.hr/ [32] u jednoj csv datoteci su posložene obitelji (nisu stvarni podaci). Dodani su im podaci OIB-a i godine rođenja te podaci koji će pomoći u daljnjem radu (šifra ulice, kombinacije OIB brojeva tko je s kim obitelj). Kad su se pripremili svi ti podaci pristupilo se popunjavanju baze podataka na Neo4j AuraDB platformi.

5.2. Punjenje baze podacima

Nekoliko je načina na koji se može popuniti baza podataka Neo4j. Prvi način bi bio putem upita bilo na Neo4j Desktop aplikaciji ili uz pomoć mrežnog resursa Neo4j AuraDB. Jedan i drugi način imaju funkcionalnost koja je ranije spomenuta, a to je Neo4j Browser. Kroz to sučelje je moguće odraditi sve potrebne upite, a moguć je donekle i grafički prikaz čvorova i relacija koji se prikazuju u skladu s upitima. Kao primjer će biti prikazano kreiranje početnih čvorova i relacije baze podataka koja se koristi za aplikaciju [33].

Prvo su se kreirali čvorovi i relacije vezano uz lokaciju. Ti upiti su se proveli na sljedeći način (prikazuje se samo dio upita pošto nema potrebe ponavljati za sva mjesta koji su bili kreirani za potrebe aplikacije):

```
CREATE
(z1:Županija {naziv_zupanije:'Grad Zagreb', id_zupanija:'zup_gr_zg'}),
(g1:Grad {naziv_grada:'Zagreb', id_grad:'gr_zg'}),
(u1:Ulica {naziv_ulice:'Borovci', id:'ul_001', id_ulica:'ul_gr_zg_01'})

CREATE
(u1)-[:PRIPADA_GRADU]->(g1),
(u1)-[:PRIPADA_ZUPANIJI]->(z1),
(g1)-[:NALAZI_SE_U]->(z1)

CREATE
pp1:PolicijskaPostaja {naziv_postaje:'Prva policijska postaja - Zagreb',
id_ustanova:'zg_1pp'})
```

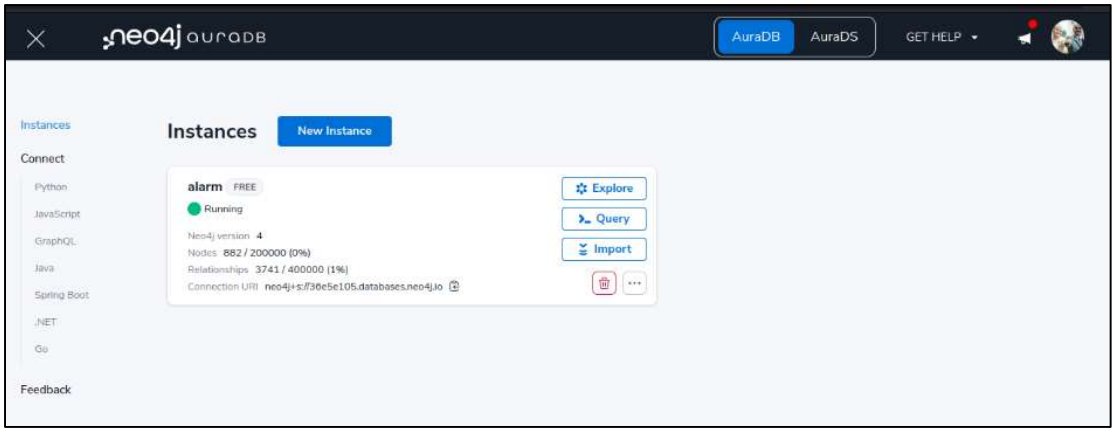
```
CREATE
(pp1)-[:LOKACIJA_PP]->(u1)

CREATE
pd:PolicijskiDjelatnik {ime:'Jure', prezime:'Pavlić',
id_djelatnik:'pol_001'}), (pd)-[:POLICIJSKI_DJELATNIK]->(pp1)
```

Na navedeni način su kreirane lokacije i službene osobe te njihove međusobne relacije. Drugi način popunjavanja baze podacima (za online servis AuraDB) je preko uvoza podataka iz csv datoteke [33]. Tako su se uveli podaci o osobama i njihove obiteljske veze. Datoteka izgleda na sljedeći način:

```
ime,prezime,OIB,godiste,obitelj,ulica,uloga_obitelj,status,skolarac,malolje
tnik,nadzor
Zorana,Čović,95723042960,1990,OB_001,ul_gr_zg_01,majka,samohrani,,,false
Maksim,Čović,34163394249,2010,OB_001,ul_gr_zg_01,dijete_1,,true,true,false
…
Irma,Burić,55932105994,2020,OB_190,ul_gr_ds_02,dijete_2,,,true,false
```

Prvi red daje imena atributima vezanim uz osobe (ime, prezime …), a poslije toga su podaci po redcima koji se međusobno odjeljuju zarezom (zbog toga csv datoteka i nosi takvo ime ekstenzije). Csv datoteka se uvodi u bazu preko funkcionalnosti koju daje AuraDB i to Import (prikazano na Slici 7).

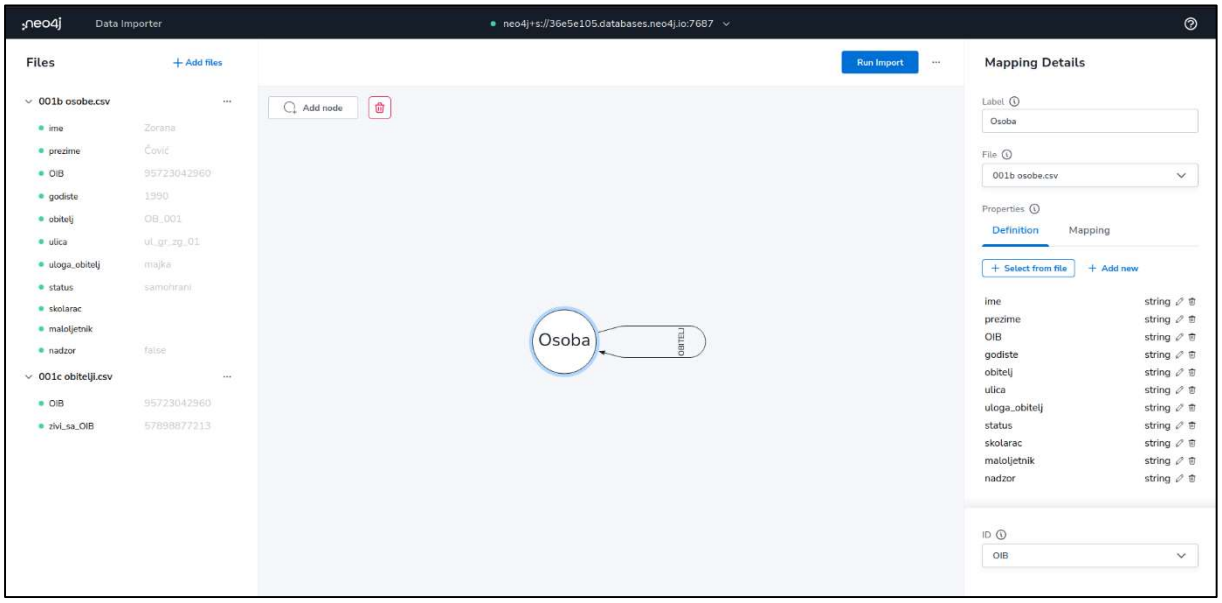


Slika 7 – Funkcija Import na AuraDB

Prije nego što se uvedu podaci svih osoba, dobra je praksa postaviti ograničenje za pojedine podatke. Kako je u ovom slučaju OIB jedinstven i predstavlja ključ potrebno je bazu zaštititi od slučajnog unosa dviju osoba s istim OIB-om. To se može odraditi uz pomoć komande **CONSTRAINT**. Isti omogućuje jedinstveno ograničenje svojstva pri čemu osigurava

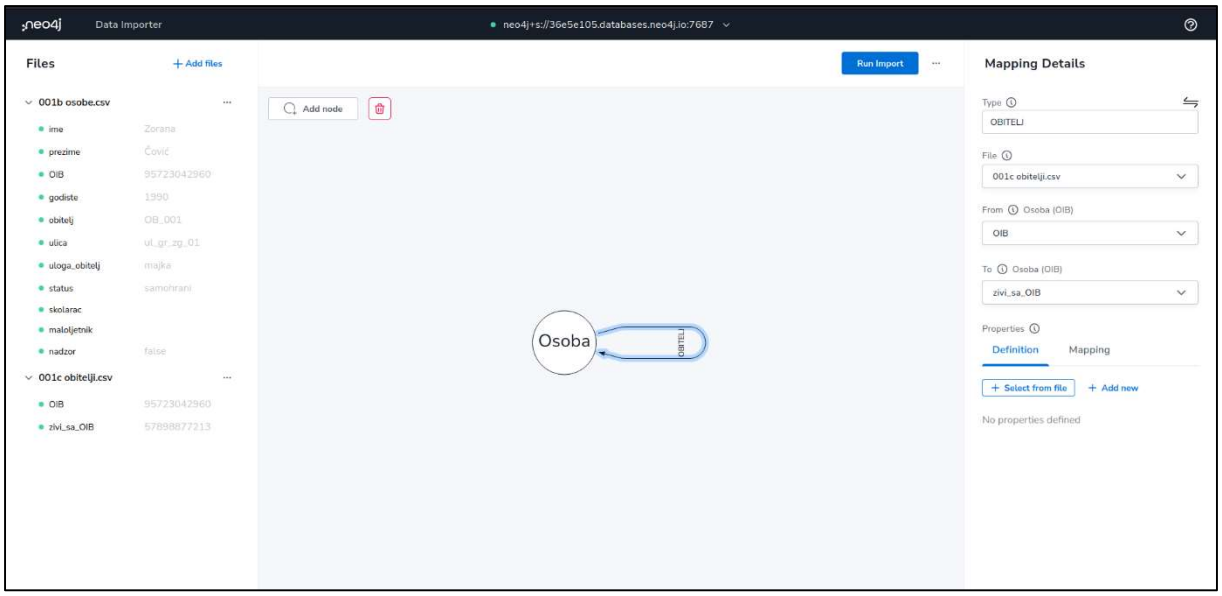
da su vrijednosti svojstva jedinstvene za sve čvorove s određenom oznakom. Ako se želi postaviti ograničenje jedinstvenih svojstava na više svojstava, kombinacija vrijednosti svojstva je jedinstvena. Jedinstvena ograničenja ne zahtijevaju da svi čvorovi imaju jedinstvenu vrijednost za navedena svojstva — čvorovi bez svih svojstava ne podliježu ovom pravilu [34]. Sljedeći redak provodi željeno ograničenje:

```
CREATE CONSTRAINT IF NOT EXISTS ON (o: `Osoba`) ASSERT o.`OIB` IS UNIQUE;
```



Slika 8 – Kreiranje čvora Osoba

Nakon što se klikne na „Import“ pojavljuje se sučelje za uvođenje podataka u bazu. Odabire se datoteka s podacima, daje se naziv čvoru koji je u ovom slučaju Osoba, odabiru se svojstva koji će se koristiti u bazi i odabire se ključ (o ovom slučaju je to OIB – Slika 8). Kako je kreirana posebna datoteka koja definira tko je s kim u obiteljskom odnosu (jedan red je OIB jednog člana obitelji, a drugi red je OIB osobe srodnika – Slika 9).



Slika 9 – Kreiranje relacije OBITELJ

Nakon uvođenja podataka o osobama, a uz pomoć svojstava Osobe i Ulice kreira se upit u Neo4j Browseru kojim se stvara relacija između osobe i ulice u kojoj živi.

```
MATCH (o:Osoba), (u:Ulica)
WHERE o.ulica=u.id_ulica
CREATE (o)-[r:ZIVI_NA]->(u)
```

Pošto se na način prikazan na Slici 9 stvorila relacija OBITELJ koja ide samo u jednom smjeru, a ipak se željela relacija koja je u oba smjera, treba se postaviti upit na sljedeći način kako bi se došlo do željenog:

```
MATCH (o:Osoba), (n:Osoba)
WHERE o.obitelj = n.obitelj
MERGE (o)-[r:OBITELJ]->(n)
```

Na ovom mjestu se pojavila nova komanda i to **MERGE**. MERGE može koristiti postojeće čvorove te ih veže ili, ako ne postoje, stvara nove podatke i povezuje ih. Tako MERGE djeluje poput kombinacije MATCH i CREATE koja dodatno omogućuje da se odredi što će se dogoditi ako su podaci upareni ili stvoreni. Na primjer, može se odrediti da graf mora sadržavati čvor za korisnika s određenim imenom. Ako ne postoji čvor s ispravnim imenom, kreirat će se novi čvor i postaviti njegovo svojstvo imena [35].

Pošto je ovim upitom stvorena i relacija na samog sebe bilo bi potrebno ukloniti istu, a to se provodi kroz sljedeći upit:

```
MATCH (o:Osoba)-[r:OBITELJ]->(o)
DELETE r
```

Kako je već ranije rečeno, polustrukturirani podaci uvijek mogu biti nadopunjeni novim podacima bez velikih intervencija u strukturu cijele baze. Dovoljno je samo jedan redak kako bi se svim osobama dodalo još jedno svojstvo. Kako je autor želio imati gumb čijim će klikom biti naznačeno kako je za pojedinu osobu upućen nadzor bilo je potrebno dodati još jedan podatak. Svim osobama u bazi je dodano svojstvo nadzora i postavljena je vrijednost na „*false*“. Upitom, a nakon klika gumba, će doći do izmjene statusa u „*true*“. Sljedeća linija koda svima naknadno dodaje još jedno novo svojstvo.

```
MATCH (o:Osoba) SET o.nadzor = FALSE
```

S ovim su se kreirali svi potrebni čvorovi i njihove međusobne relacije izuzev događaja koji se tek trebaju kreirati i relacije događaja s osobama, institucijama i djelatnicima institucija. Isti će se kreirati nakon izrade aplikacije kroz nju samu, a u cilju provjere funkcionalne ispravnosti aplikacije.

Još dva korisna upita su korištena tijekom izrade baze podataka i popunjavanja iste podacima, ali nisu dio baze podatka koja se nalazi u pozadini aplikacije. Prvi upit stvara grafički prikaz sheme i iz nje se može vidjeti jesu li se slučajno potkrale greške tijekom kreiranja čvorova i međusobnih relacija [36].

```
CALL db.schema.visualization
```

Drugi upit provodi brisanje svih čvorova i relacija. Neo4j onemogućuje slučajno brisanje čvora koji ima nekakvu relaciju, stoga je potrebno prvo ukloniti sve veze uz pomoć DETACH komande [37].

```
MATCH (n)
DETACH DELETE n
```

6. Izrada aplikacije

Nakon modeliranja baze podataka i popune iste s podacima može se prijeći na izradu aplikacije. Za potrebe ovog završnog rada odabrana je Node.js platforma. Node.js je višeplatformno radno okruženje otvorenog koda izgrađeno na temeljima JavaScripta. Isti se pokreće u Google Chrome V8 pokretaču izvan preglednika. Zbog toga je Node.js vrlo učinkovit [38]. Isti se temelji na JavaScript jeziku, ali ta platforma nudi više od JavaScripta. Node.js omogućava programiranje na korisničkoj strani (što je slučaj kod JavaScripta), ali i na serverskoj strani [39].

6.1. Iniciranje Node.js aplikacije

Prvi korak u izradi aplikacije bi bio instalacija radnog okruženja Node.js. Na web stranici https://nodejs.org/en/ ponuđene su dvije opcije. Preuzimanje stabilne verzije koja je preporučena za rad te zadnje verzije kod koje su moguće greške. Kada se započela izrada aplikacije, zadnja stabilna verzija radnog okruženja je bila 16.15.1. Nakon preuzimanja provede se jednostavna radnja instalacije radnog okruženja. Kako je za potrebe izrade aplikacije bilo potrebno znati koja se verzija koristi dobro je znati da se do tog podatka dolazi komandom „node --version“ u naredbenom retku [40].

Izuzetno koristan dodatak Node.js radnom okviru je Npm. Npm je upravitelj paketima za Node.js. Stvoren je 2009. kao projekt otvorenog koda kako bi se pomoglo razvojnim programerima koji rade u JavaScript jeziku. Sustav služi za lakše dijeljenje paketa kodova. Da bi se sustav instalirao na računalo potrebno je u linijskom retku upisati „sudo apt install npm“ (za Linux korisnike) [41].

Nakon instalacije Node.js radnog okruženja i Npm upravitelja paketa može se započeti s kreiranjem aplikacije. Potrebno je kreirati mapu u kojoj će se nalaziti dokumenti i druge mape projekta. U toj mapi se pokreće terminal i inicira se aplikacija komandom „npm init“. Tijekom inicijalizacije može se odabrati ime aplikacije, ulazna datoteka (nudi index.js, ali može se dati bilo koje ime) i još neki podaci koji će biti pospremljeni u jednu novu datoteku pod imenom package.json [42].

Datoteka package.json se sastoji od postavki projekta i popisa ovisnosti o Npm paketima. U prilogu završnog rada može se vidjeti sadržaj package.json datoteke nakon instaliranja svih potrebitih dodataka koji su omogućili rad aplikacije.

Sljedeći korak bi bio kreiranje datoteke aplikacije, koja je u ovom slučaju imenovana kao app.js. Ona je glavna ulazna i izlazna točka i u njoj se nalazi sva programska logika. Prije nego se počne popunjavati datoteka aplikacije izuzetno korisno je instalirati dodatak Express.

Express je minimalistički i fleksibilan Node.js okvir za web aplikacije koji pruža skup značajki potrebnih jednoj web i mobilnoj aplikaciji. Kako se uz pomoć Node.js odrađuje back-end, potreban je dodatak kojim će se sačiniti front-end, a tu u pomoć uskače Express [43].

Na web adresi https://www.npmjs.com/package/express se mogu pronaći detalji o tom dodatku, a minimalistički program „Hello World“ bi izgledao na sljedeći način [43]:

```
const express = require('express')
const app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})

app.listen(3000)
```

Ovako sročena aplikacija na adresi localhost:3000, nakon pokretanja aplikacije u terminalu uz pomoć komande „node app.js“, prikazuje „Hello World“. Osim komande „node“ može se koristiti i „nodemon“ ako je ista instalirana (niže je objašnjena).

Dodaci koji su se koristili tijekom izrade aplikacije su sljedeći:

- ejs - (*Embedded JavaScript Templating*) jedan je od najpopularnijih pokretača predložaka za JavaScript. Kako samo ime kaže, isti omogućuje ugrađivanje JavaScript koda u jezik predloška koji se zatim koristi za generiranje HTML-a [44]
- express - već je ranije objašnjen
- express-openid-connect - implementacija softvera za prijavu u web aplikaciju pomoću OpenID Connecta [45]
- neo4j-driver - službeni Neo4j drajver za JavaScript [46]
- nodemon - nodemon je alat koji pomaže pri razvoju aplikacija temeljenih na Node.js automatskim ponovnim pokretanjem aplikacije kada se otkriju promjene datoteke [47]
- opener - otvara web preglednik po pokretanju aplikacije [48]

6.2. Programska logika app.js datoteke

Glavno mjesto gdje su svi ulazi i izlazi iz sačinjene aplikacije je datoteka app.js. Ona u sebi sadrži nekoliko funkcionalnih cjelina. Prvi dio koji je prikazan u nastavku učitava potrebite dodatke (prvi pasus), podatak za pokretanje aplikacije u pregledniku (drugi pasus), podatke za okruženje (treći pasus), postavlja radni okvir (express - četvrti pasus), određuje postavke „stroja“ koji će ga pokretati, gdje se nalaze statični dokumenti kako bi se znalo s njima rukovati, način rukovanja s podacima i dodatak koji odrađuje dio s autorizacijom i autentifikacijom (peti

pasus). Na kraju se određuje port na kojoj će aplikacija dati izlaz (šesti pasus) te izvozi podatke od sebe prema ostalim datotekama (sedmi pasus).

```
const express = require('express');
const neo4j = require('neo4j-driver');
const bodyParser = require('body-parser');
const { auth } = require('express-openid-connect');

const opener = require('opener');
opener('http://localhost:3000');

const config = {
  authRequired: false,
  auth0Logout: true,
  secret:
"lčklkjslkdfnvpoisevosidjgćlskvdćydjvčlkevnćvmčylvčvlčmsčlcvsdmvčdfkjv",
  baseURL: "http://localhost:3000",
  clientID: "YvnwhhthjSBkMGY4dJNcbZLWLJAz1R9T",
  issuerBaseURL: "https://dev-xhzjybdg.us.auth0.com",
};

const app = express();

app.set('view engine', 'ejs');
app.use(express.static ('public'));
app.use(express.static ('views'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: false}));
app.use(auth(config));

const port = 3000;
app.listen(port);

module.exports = app;
```

U nastavku se provodi spajanje s bazom podatka kako bi se mogli postavljati upiti prema njoj.

```
var driver = neo4j.driver('neo4j+s://36e5e105.databases.neo4j.io',
neo4j.auth.basic('neo4j', 'Eg7O6ak1ErjFPeCwsi-IMcQds9gWKcQDDUx8jBN9mco'));
session = driver.session();
```

Aplikacija ima nekoliko GET (prijem podataka) i POST (slanje podataka) dijelova pa će biti opisani samo po jedan primjer od svakog.

6.2.1. Prijem podataka

Primjer koji se nalazi niže je iz aplikacije i njegova funkcija je dohvat određenih podatka koji se prikazuju u dokumentu „lista“. Pokreće se upit koji obuhvaća sve osobe koje imaju obiteljsku vezu. Klauzulom WHERE za prikaz se filtriraju osobe koje su mlađe od 18 godina te nije poslan nadzor. Opcionalnim upitima (OPTIONAL MATCH se koristi kas se očekuje da u nekim slučajevima može doći do odgovora s vrijednošću NULL) računa koliko je slučajeva u obitelji koji imaju zakonske te socijalne probleme te se za maloljetnu osobu računa broj zdravstvenih i školskih problema. Svi ti brojevi se zbrajaju u jedan zbroj te se lista formira u skladu s tim brojem. Na vrhu liste je onaj koji ima najviše bodova, a upitom se ograničava broj osoba za prikaz na 10. Dio podataka tog upita (ime, prezime, OIB i nadzor) se smještaju u polje s podacima osoba te se nakon toga polje podataka šalje prema dokumentu „lista“. Tu listu mogu vidjeti samo osobe koje su prošli kroz autentifikaciju. Nakon toga se zatvara sesija prema bazi podataka i hvataju se pogreške koje se ispišu u konzoli.

```
app.get('/lista', function(req, res){
  session
    .run(
'MATCH (o1:Osoba)-[r1:OBITELJ]->(o2:Osoba)
WHERE (date().year - toInteger(o1.godiste)) < 18 AND o1.nadzor = "false"
OPTIONAL MATCH (o1)-[r1]->(ps:`ProblemŠkola`)
OPTIONAL MATCH (o1)-[r2]->(zdp:`ZdravstveniProblem`)
OPTIONAL MATCH (o1)-[r3]->(o2:Osoba)-[r4]->(zp:ZakonskiProblem)
OPTIONAL MATCH (o1)-[r5]->(o2:Osoba)-[r6]->(sp:SocijalniProblem)
RETURN o1, count(distinct r1)+count(distinct r2)+count(distinct
r4)+count(distinct r6) AS Bodovi
ORDER BY Bodovi
DESC LIMIT 10')
    .then(function(result){
        var osoba = [];
        result.records.forEach(function(record){
            osoba.push({
                ime: record._fields[0].properties.ime,
                prezime: record._fields[0].properties.prezime,
                OIB: record._fields[0].properties.OIB,
                nadzor: record._fields[0].properties.nadzor
            });
        });
        res.render('lista', { title: 'ALARM',
            osobe: osoba,
            isAuthenticated: req.oidc.isAuthenticated(),
            user: req.oidc.user,
        });
        session.close;
    })
```

```
    .catch(function(err){
        console.log(err);
    });
});
```

6.2.2. Slanje podataka

Kod slanja podataka potrebno je obuhvatiti tražene podatke prije slanja, a to se odrađuje putem forme. U nastavku je primjer unosa podataka od strane policijskog djelatnika (sama forma će biti prikazana u sljedećem potpoglavlju). U kodu je vidljivo kako se iz forme čije je ime „/dogadjaj/dodaj_policija“ obuhvaćaju podaci POST metodom koji se smještaju u varijable dog_OIB, id_ustanova, vrijeme i vrsta_dogadjaj. Nadalje, u kodu je vidljivo kako se ti podaci svrstavaju u upit gdje se prvo traži osoba, policijski djelatnik i policijska postaja te se nakon toga kreira čvor ZakonskiProblem kojeg opisuju svojstva „vrijeme“ i „vrsta događaja“. Potom se kreiraju veze između osobe koja je počinila događaj i čvora koji opisuje događaj, te se dodatno, za potrebe moguće kasnije detaljnije analize stvaraju veze između policijskog djelatnika i događaja te policijske postaje i događaja.

```
app.post('/dogadjaj/dodaj_policija', function(req, res){

  var dog_OIB = req.body.dog_OIB;
  var id_ustanova = req.body.id_ustanova;
  var vrijeme = req.body.dog_vrijeme;
  var vrsta_dogadjaj = req.body.vrsta_dogadjaj;

      session
      .run(
'MATCH (o:Osoba{OIB: $dog_OIB}), (d:PolicijskiDjelatnik{id_djelatnik:
"pol_001"}), (i:PolicijskaPostaja{id_ustanova: $id_ustanova})
MERGE (o)-[r1:ZAKONSKI]->(p:ZakonskiProblem{vrijeme: $vrijeme,
vrsta_dogadaja: $vrsta_dogadjaj})
MERGE (d)-[r2:POLICIJA_EVIDENCIJA]->(p)
MERGE (p)-[r3:POLICIJA_LOKACIJA]->(i)
RETURN o, d, i, p', {dog_OIB: dog_OIB, id_ustanova: id_ustanova, vrijeme:
vrijeme, vrsta_dogadjaj: vrsta_dogadjaj, })
      .then(function(result){
        res.redirect('/unos_dogadjaja_policija');
        session.close;
        })
      .catch(function(err){
        console.log(err);
        });
      });
```

6.2.3. Unos podataka

Za unos podataka se rabe forme. U nastavku će biti opisan način unosa. Na početku koda se može primijetiti jedan dio sačinjen uz pomoć ejs modula. Iz njega je vidljivo kako se autentificirane osobe koje nemaju e-mail adresu na serveru mup.hr preusmjeravaju na početnu stranicu. Taj dio služi autorizaciji korisnika koji će biti objašnjen u sljedećem potpoglavlju.

Iz HTML koda forme je vidljivo kako policijski djelatnik unosi OIB osobe za koju se veže određeni događaj. Pored toga, unosi se datum događaja, područje policijske postaje na kojoj se događaj desio te tip događaja. U aplikaciji se nalaze samo po dva tipa događaja po vrsti događanja (policija, zdravstvo, socijalna služba i školstvo) u cilju prikaza mogućnosti. Naravno, u aplikaciji koja bi se koristila u stvarnim uvjetima tipova događanja bi bilo svakako više.

```
<html lang="hr">

    <% if (user.email.split('@').reverse()[0] !== "mup.hr") { %>
        <script>window.location.href = "/";</script>
    <% } %>
    <%- include("./partials/head.ejs") %>
<body>
    <%- include("./partials/nav.ejs") %>
    <div>
        <h1>Dodaj događaj</h1><br>

            <form action="/dogadjaj/dodaj_policija" method="post">
            <label for="dog_OIB">OIB osobe:</label>
            <input type="text" name="dog_OIB" size="50"
required="required"><br>
            <label for="dog_vrijeme">Vrijeme događaja:</label>
            <input type="date" name="dog_vrijeme" required="required"><br>
            <label for="id_ustanova">Ustanova: </label>
                <select name="id_ustanova">
                    <option value="zg_1pp">1 (prva) policijska postaja -
Zagreb</option>
                    <option value="sam_2pp">2 (druga) policijska postaja -
Samobor</option>
                    <option value="ds_3pp">3 (treća) policijska postaja -
Dugo Selo</option>
                </select><br>
            <label for="vrsta_dogadjaj">Vrsta događaja: </label>
                <select name="vrsta_dogadjaj">
                    <option value="Posjedovanje opojnih
sredstava">Posjedovanje opojnih sredstava</option>
                    <option value="Obiteljsko nasilje">Obiteljsko
nasilje</option>
                </select><br><br>
                <button type="submit">Pošalji</button>
```

```
    </div>
    <%- include("./partials/footer.ejs") %>
</body>
</html>
```

6.2.4. Kreiranje ruta (usmjeravanje)

Usmjeravanje se odnosi na određivanje načina na koji aplikacija odgovara na zahtjev klijenta prema određenoj krajnjoj točki, što je URI i specifična metoda HTTP zahtjeva (GET, POST ili neka druga metoda). Svaka ruta može imati jednu ili više funkcija rukovatelja, koje se izvršavaju kada se ruta podudara. Definicija rute ima strukturu „app.METHOD(PATH, HANDLER)“. App je express instanca, METHOD je metoda HTTP zahtjeva pisana malim slovima, PATH je putanja na poslužitelju, a HANDLER je funkcija koja se izvršava kada se ruta podudara. Bez navedenih ruta nije moguće pristupiti traženim web dokumentima [49].

```
app.get('/', (req, res) => {
  res.render('index', { title: 'O aplikaciji',
  isAuthenticated: req.oidc.isAuthenticated(),
  user: req.oidc.user,
});
});
```

Iz ovog dijela koda je vidljivo kako ovoj ruti mogu pristupiti samo osobe koje su se autentificirale.

6.2.5. Brisanje osobe s liste

Lista koja daje popis desetoro najugroženije djece je jedan od konačnih proizvoda aplikacije (grafički prikaz slučaja je drugi). Međutim, ako se uputi nadzor prema obitelji djeteta s liste i po stavljanju obitelji u program praćenja bilo bi potrebno dijete maknuti s liste. Kako bi se to učinilo prvo se u tablici u kojoj se nalazi popis djece (lista.ejs) u svakom retku dodala još jedna ćelija s formom koja POST metodom šalje broj OIB-a djeteta.

```
        <% osobe.forEach(function(osoba){ %> <% if (osoba.nadzor ==
'false') {
        %>
        <tr>
          <td><%= osoba.ime %></td>
          <td><%= osoba.prezime %></td>
          <td><%= osoba.OIB %></td>
          <td style="text-align: center">
            <form action="lista" method="post">
              <input type="hidden" name="nadzor_OIB" value="<%= osoba.OIB
%>" />
```

```
                <button type="submit">Poslati nadzor</button>
              </form>
            </td>
          </tr>
          <% } %> <% }) %>
```

Nakon što službenik klikne gumb, djetetov OIB se upućuje POST metodom prema datoteci app.js koja preko upita mijenja sadržaj svojstva „nadzor“ iz „false“ u „true“. Odmah nakon odrađenog upita provodi se preusmjerenje prema stranici lista na kojoj se nalazi desetoro najugroženije djece, ali ovaj puta prikaz je bez djeteta za koje je upućen nadzor.

```
app.post('/lista', function(req, res){

    var nadzor_OIB = req.body.nadzor_OIB;

    session
        .run('MATCH (o:Osoba{OIB: $nadzor_OIB}) SET o.nadzor = "true"
RETURN o', {nadzor_OIB: nadzor_OIB})
        .then(function(result){
            res.redirect('/lista');
            session.close;
        })
        .catch(function(err){
            console.log(err);
        });
});
```

6.2.6. Autentifikacija i autorizacija

Na početku se treba napraviti razlika između pojmova autentifikacije i autorizacije. Često se ta dva pojma u uporabi miješaju iako imaju različita značenja. Toj zbrci najvjerojatnije pridonosi činjenica kako autorizacija ovisi o autentifikaciji i da su vezani pojmovi. Za autentifikaciju se može reći kako je to čin identificiranja korisnika ili uređaja. Autorizacija je čin dozvole ili zabrane pristupa resursu nekom autentificiranom korisniku ili uređaju [50].

Pri izradi ove aplikacije korišten je sustav kompanije auth0. Auth0 je fleksibilno rješenje za dodavanje usluga provjere autentičnosti i autorizacije aplikacijama. Time se izbjegavaju dodatni troškovi izgradnje sustava sigurnosti, štedi se vrijeme za izradu istog te se umanjuje rizik koji dolazi s izgradnjom vlastitog rješenja za autentifikaciju i autorizaciju korisnika [51].

Kako se besplatnim planom uporabe auth0 sustava dobije samo mogućnost registracije 7.000 korisnika i jedino je moguće koristiti opcija autentifikacije bilo je potrebno iznaći rješenje kako provesti autorizaciju na aplikaciji. Uz pomoć funkcije „split“ JavaScripta i upravljanjem

e-mail adresama uspjela se postići i autorizacija na aplikaciji. Autorizacija se odradila kroz dva koraka. U prvom koraku se u navigaciji postavilo pravilo koja e-mail domena ima pravo pristupa određenom dijelu aplikacije. Kako bi se izbjegla mogućnost neautoriziranog pristupa, na početku svakog web dokumenta je postavljen logički uvjet za pristup resursu ili u suprotnom se usmjerava na glavnu stranicu. Taj drugi dio je već prikazan ranije u radu, a prvi dio se vidi u sljedećem dijelu koda aplikacije.

```
<nav>
  <div class="site-title">
    <a href="/"><h1>Alarm - FOI</h1></a>
    <% if(user) { %>
    <p>Dobrodošli <%=user.name %></p>
    <% } else { %>
    <p>Neregistrirani korisnik</p>
    <% } %>
  </div>
  <ul>
    <a href="/">O aplikaciji</a>
    <% if(user) { %> <% if (user.email.split('@').reverse()[0] ===
"mup.hr") {
    %>
    <a href="/unos_dogadjaja_policija"> | Unos događaja - Policija</a>
    <% } else if (user.email.split('@').reverse()[0] === "czss.hr") { %>
    <a href="/unos_dogadjaja_socijalna"> | Unos događaja - CZSS</a>
    <% } else if (user.email.split('@').reverse()[0] === "miz.hr") { %>
    <a href="/unos_dogadjaja_zdravstvo"> | Unos događaja - Zdravstvo</a>
    <% } else if (user.email.split('@').reverse()[0] === "skole.hr") { %>
    <a href="/unos_dogadjaja_skola"> | Unos događaja - Škola</a>
    <% } else { %>
    <a href="/unos_dogadjaja_ostali"> | Unos događaja - Ostali</a>
    <% } %> <% } %> <% if(user) { %>
    <a href="/stanje"> | Ukupna lista</a>
    <a href="/lista"> | Alarm lista</a>
    <% } %> <% if(!isAuthenticated) { %>
    <a href="/login/"> | Login</a>
    <% } else { %>
    <a href="/logout/"> | Logout</a>
    <% } %>
  </ul>
</nav>
```

6.3. Grafički prikaz

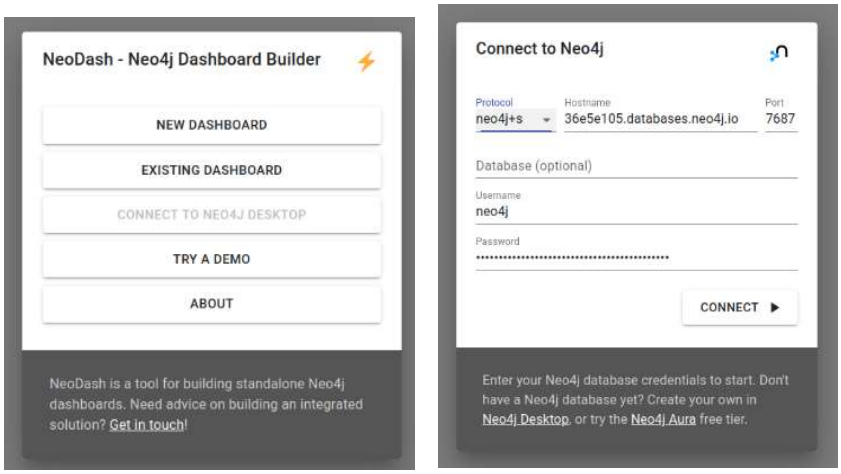
Postoji izreka da jedna slika vrijedi kao tisuću riječi. Kako je Neo4j grafička baza podataka, logično je za očekivati da se rezultati prikažu na grafički način. Osim što grafički prikaz efikasnije opisuje određene situacije, osobe koje unose podatke u aplikaciju imaju

očekivanja kako će se vidjeti nekakav rezultat njihova angažmana. Zbog toga se aplikaciji dodaje dio koji na vizualan način prezentira podatke.

Neo4j je podržan od strane mnogih dodataka za vizualizaciju podataka. Neki od njih su dio Neo4j (Browser - alat za razvoj, Bloom - istraživanje i analiza), a neki su izrađeni od strane različitih tvrtki i skupina ljudi koji su željeli iskoristiti opcije Neo4j (Neovis.js, Popoto.js, KeyLines, CytoScape, yWorks, Linkurious Enterprise, GraphAware Hume, Kineviz GraphXR, Graphistry, Tom Sawyer's Perspectives, Graphileon, Charts i NeoDash). Isti se mogu svrstati u alate za razvoj, istraživanje, analizu i izvješćivanje [52].

Za potrebe ovog rada koristi se NeoDash. NeoDash je aplikacija koja na brz način može izraditi nadzorne ploče iz Neo4j podataka, a za potrebe web aplikacija. Nakon spajanja na bazu podataka generiraju se izvješća iz rezultata Cypher upita. Rezultati upita mogu se prikazati kao tablice, grafikoni, stupčasti grafikoni itd., a korisnici mogu interaktivno odabrati parametre za izvješća [52].

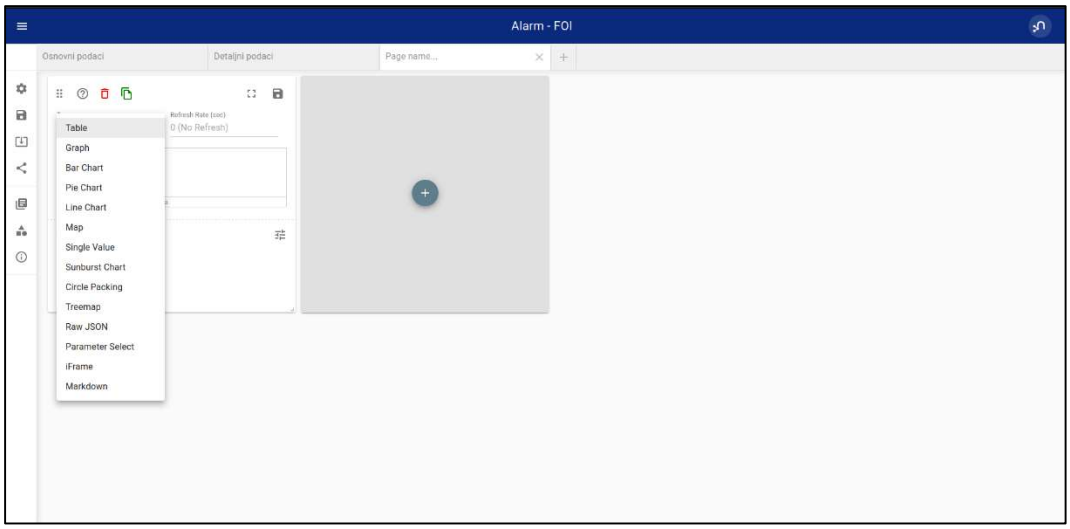
Kako bi se započela izrada grafičkog prikaza potrebno je otići na adresu http://neodash.graphapp.io/. Moguće je izraditi novu ploču, učitati staru, spojiti se s Neo4j Desktop aplikacijom i isprobati Demo prikaz opcija koje nudi NeoDash. Odabirom stvaranja nove ploče dolazi se do sučelja gdje je potrebno unijeti podatke kako bi se aplikacija spojila na bazu podataka. To je vidljivo na Slici 10 [53].



Slika 10 - Početno sučelje NeoDash aplikacije [53]

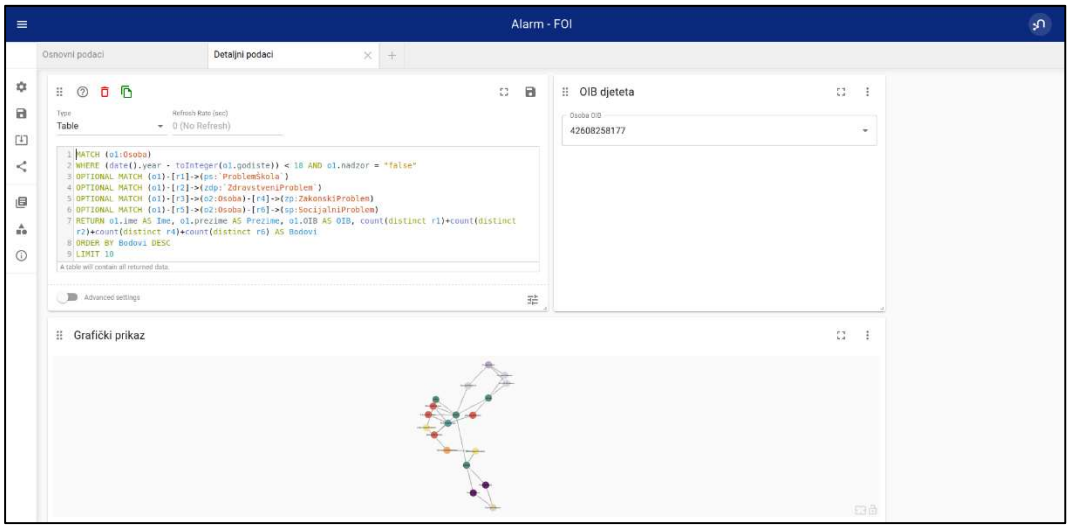
Nakon upisa podataka potrebnih za spajanje na bazu podataka može se pristupiti izradi ploče. Podaci se mogu organizirati u grupe podataka do kojih se dolazi klikom na jedan od jahača (na Slici 11 se vide dva jahača „Osnovni podaci" i „Detaljni podaci"). Na istoj slici je vidljivo koji se sve oblici prikaza mogu sačiniti na ploči. Pored tablice i grafa, moguće je kreirati

razne grafikone, geografske karte, izbornik parametra, iFrame, okvir za tekst s objašnjenjem (Markdown).



Slika 11 - Opcije kreiranja pločice na NeoDash

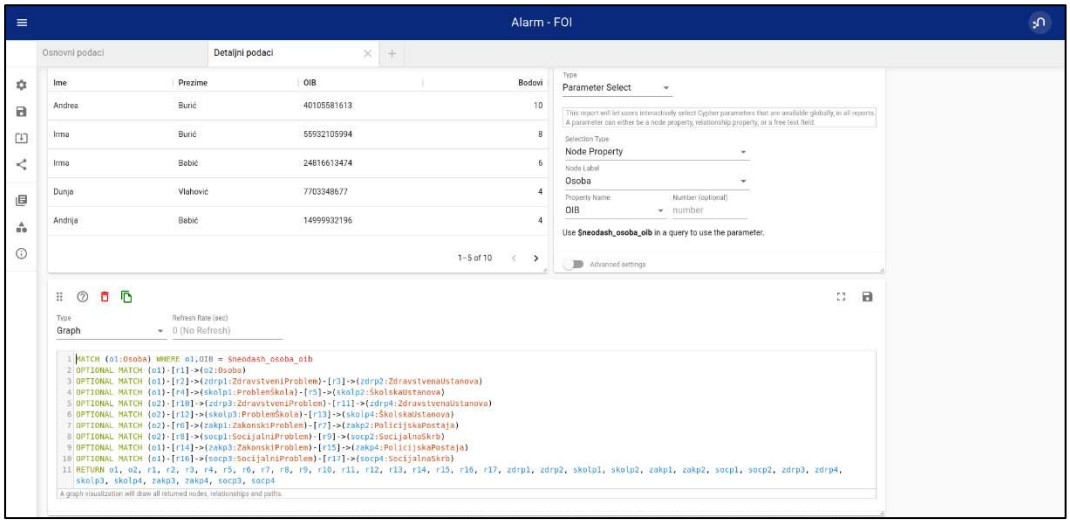
Kako bi se dobila tablica koja stvara rang listu deset najugroženije djece, odabire se prikaz tipa tablica i unosi upit vidljiv na Slici 12. Kod za sve prikaze će se nalaziti u prilogu ovog rada.



Slika 12 - Upit za kreiranje tablice na ploči NeoDash

Jedna zanimljiva opcija, koja daje svojstvo interaktivnosti prikaza je „*Parameter selected*“. Potrebno je stvoriti dvije nove pločice. Jedna će služiti unosu parametra, a druga će prikazivati podatke vezane uz odabrani parametar. Kako je OIB ključ kojeg koristi baza

podataka, isti će se koristiti i za prikaz. Kod pločice za parametar bira se svojstvo čvora, zatim koji čvor će biti izabran te svojstvo koje će se odabirati. NeoDash stvara parametra $neodash_osoba_oib kojeg je potrebno koristiti na povezanoj pločici (Slika 13). Kada se odabere drugo svojstvo, ploča generira novi grafički prikaz vezan uz parametar.



Slika 13 - Uporaba parametra kod interaktivnog prikaza

Kako se ne bi izgubio sav rad na NeoDash ploči, moguće je pospremiti učinjeno na dva načina. Moguće je pohraniti na online server na kojem je smještena i baza podataka (to stvara dodatni čvor u bazi). Druga opcija je pospremanje u json datoteku na osobnom računalu. Kako bi svi koji pristupe aplikaciji imali željeni pregled bez potrebe učitavanja lokalne datoteke, praktičnije je postavke pospremiti u bazi s kojom se radi. Jedino što kod prikaza sheme je vidljiv dodatni čvor, ali isti ne umanjuje vrijednost prikaza sheme u velikoj mjeri tako da je to prihvatljiv ustupak.

6.4. Generiranje izvršne datoteke

Nakon što se napisao cijeli program sačinjena je izvršna datoteka. Za potrebe ovog rada korišten je dodatak pkg. Pkg radi u sučelju naredbenog retka i omogućuje pakiranje Node.js projekta u izvršnu datoteku koja se može pokrenuti čak i na uređajima bez instaliranog Node.js. Pri tome se kroz opcije može podesiti vrsta izvršne datoteke. Omogućava izradu za operativne sustave Windows, Linux i MacOS [54].

Potrebno je prethodno u package.json datoteci provesti određena izmjene. Dodaje se opcija bin, i pkg pri čemu ova druga daje do znanja dodatku koje bi se sve mape trebale obuhvatiti pakiranjem. Komanda koja se pokreće u terminalu je pkg . (točka na kraju, umjesto nje se može napisati package.json).

7. Uporaba aplikacije

Kako bi aplikacija bilo što jednostavnija za uporabu, sačinila se izvršna datoteka. Pokretanjem iste u pozadini se pokreće lokalni server i podiže se web preglednik s adresom http://localhost:3000 na kojoj je vidljiva aplikacija. Kako nitko nije prijavljen u aplikaciju prikazuju se samo početna stranica i postoji mogućnost prijave (Slika 14).

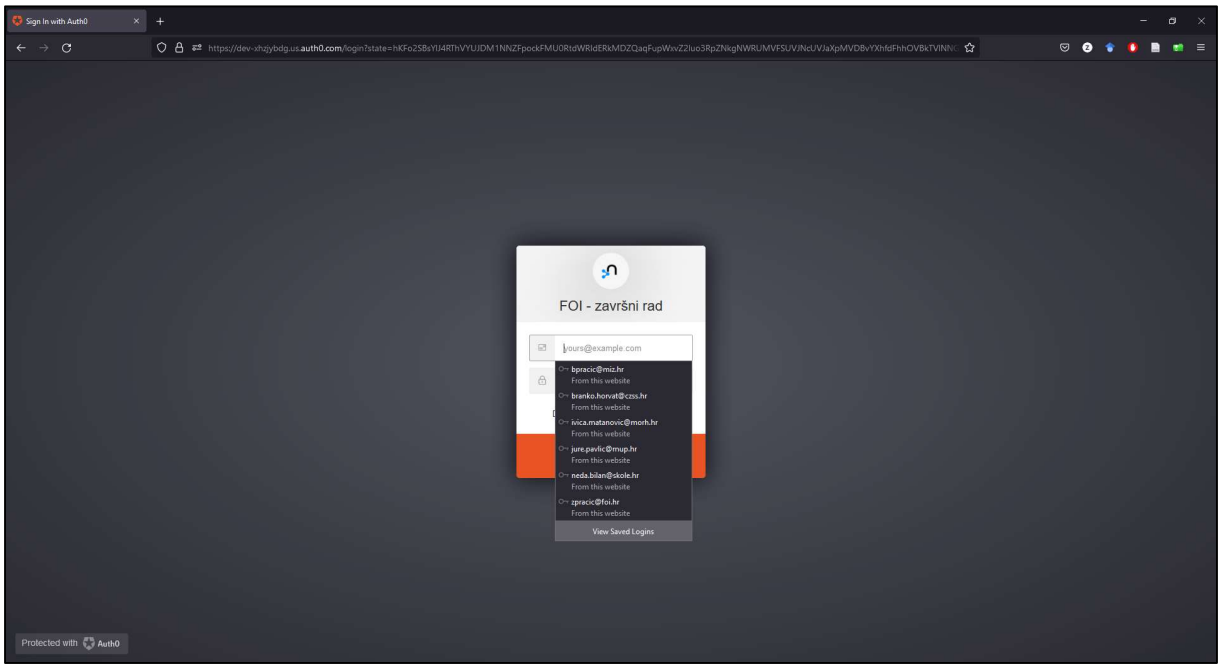
Slika 14 - Početni zaslon aplikacije

7.1. Korisnici sustava

Klikom na LOGIN pojavljuje se forma za prijavu koja u pozadini ima auth0 sustav. Na Slici 15 su vidljive različite adrese za prijavu pošto je aplikacija zamišljena za uporabu u više institucija, a iste su korištene tijekom unosa podataka u bazu podataka. E-mail adrese koje su se rabile pri izradi aplikacije nisu stvarne, a koristile su se sljedeće adrese:

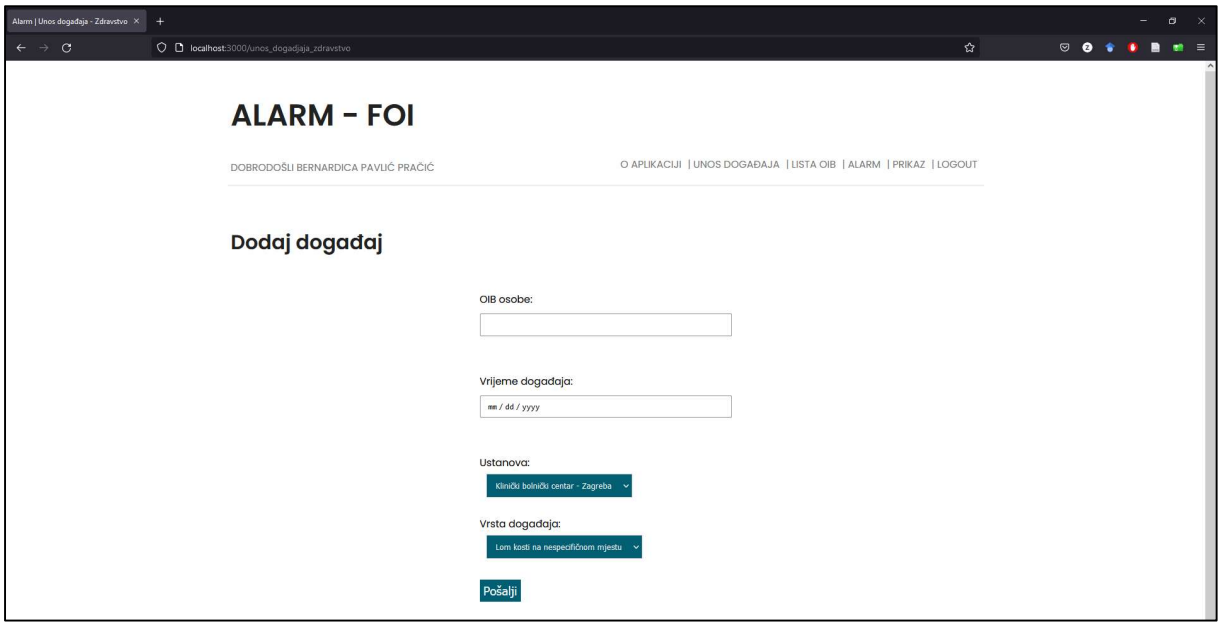
- bpracic@miz.hr - Ministarstvo zdravstva RH
- branko.horvat@czss.hr - Centar za socijalnu skrb
- ivica.matanovic@morh.hr - Ministarstvo obrane RH
- jure.pavlic@mup.hr - Ministarstvo unutarnjih poslova RH
- neda.bilan@skole.hr - Ministarstvo znanosti i obrazovanja.

Adresa Ministarstva obrane je dodana kao primjer mogućnosti pregleda rezultata, rang liste i grafičkog prikaza, bez mogućnosti unosa u samu aplikaciju. Password za sve korisnike je isti i glasi: foi-123456.



Slika 15 - Forma za prijavu u aplikaciju

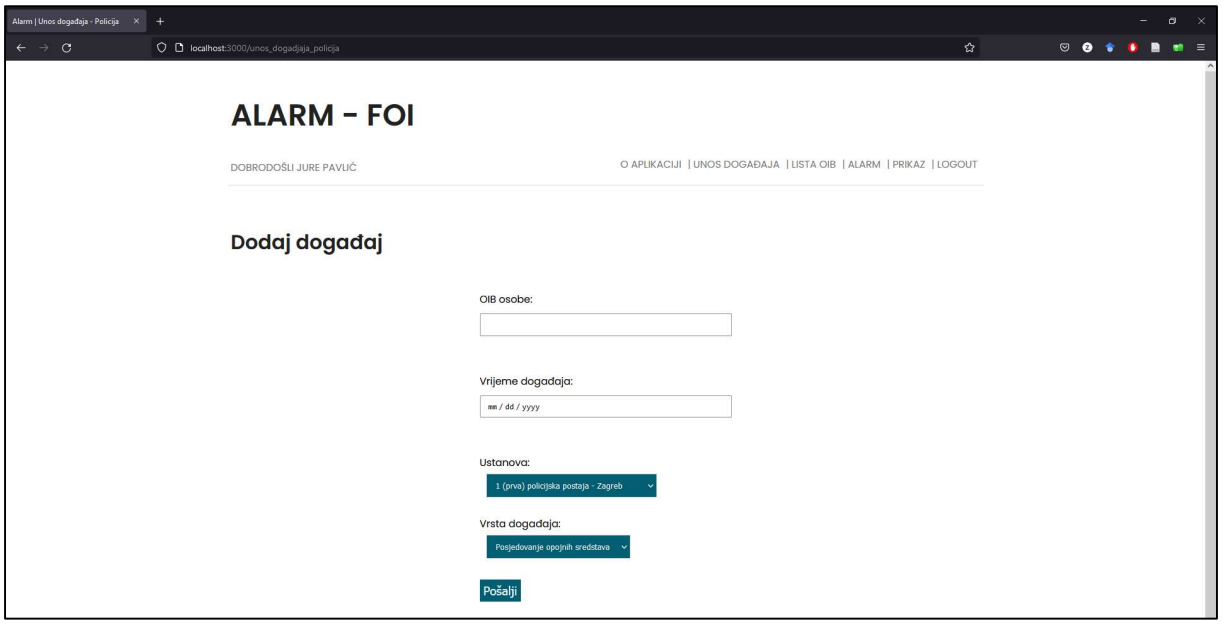
Slika 16 prikazuje kako izgleda aplikacija nakon logiranja u nju.



Slika 16 - Forma za unos događaja (zdravstvo)

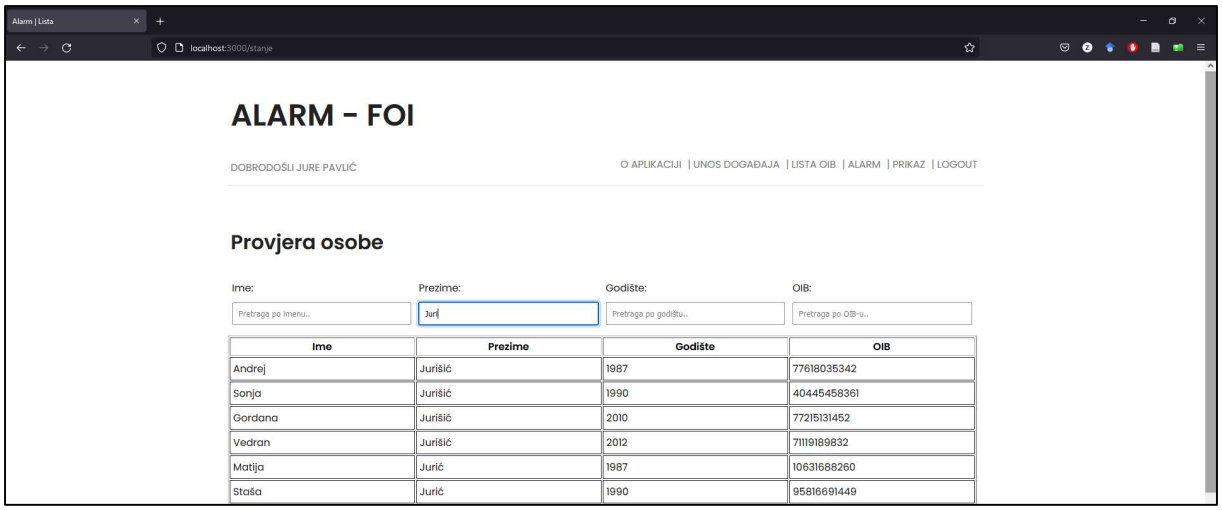
7.2. Unos podataka

U formu za unos podatka o događaju unose se podaci, ovisno o instituciji kojoj pripada osoba koja se logirala u aplikaciju. U ovom slučaju se radi o osobi koja radi u zdravstvenom sustavu. Aplikacija izgleda identično za unos podataka korisnika iz neke druge institucije pa tako osoba koja je djelatnik policije, nakon logiranja u aplikaciju, vidi prikaz sa Slike 17. Vidljivo je kako su ustanove drugačije (policijske postaje) kao i vrste događaja.



Slika 17 - Forma za unos događaja (policija)

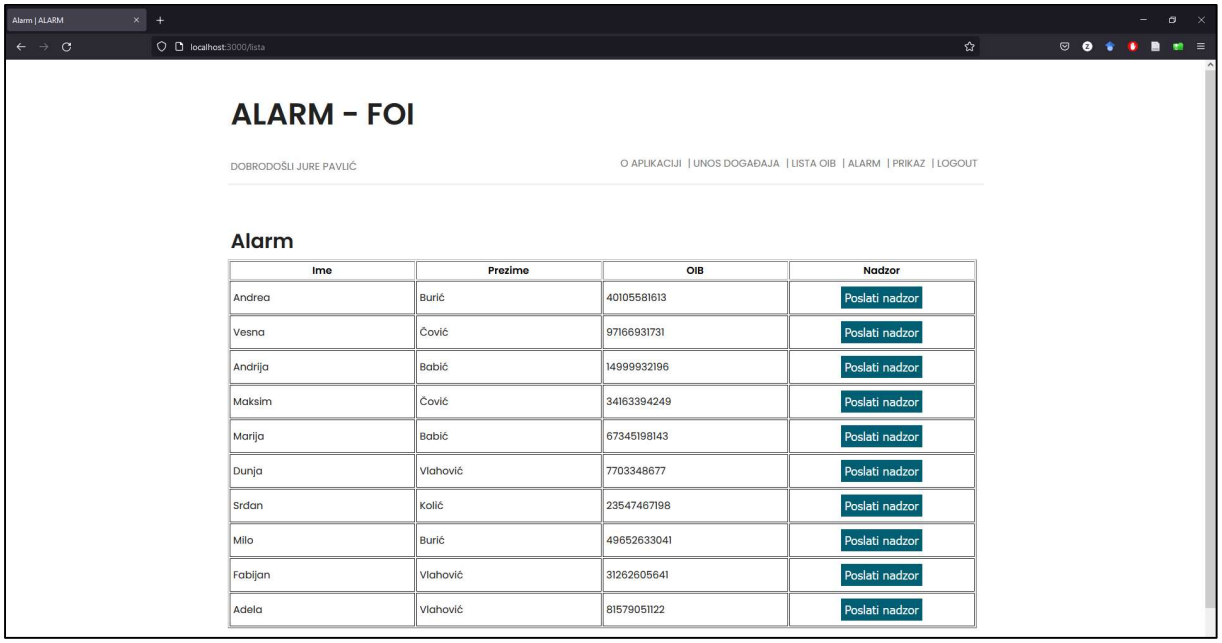
Osnovni podatak koji se unosi u aplikaciju je OIB. Nakon toga se unosi datum događaja, ustanova na čijem se području desio događaj te sam događaj. Kako bi se provjerio OIB, aplikacija ima izbornik za pretraživanje osoba te se podaci mogu pretraživati po imenu, prezimenu, godištu i OIB-u. Na Slici 18 se vidi kako su osobe pretraživane pomoću prezimena i nakon upisivanja u okvir „Juri“ prikazuju se prezimena koja počinju s tom ključnom riječi.



Slika 18 - Pretraživanje osoba

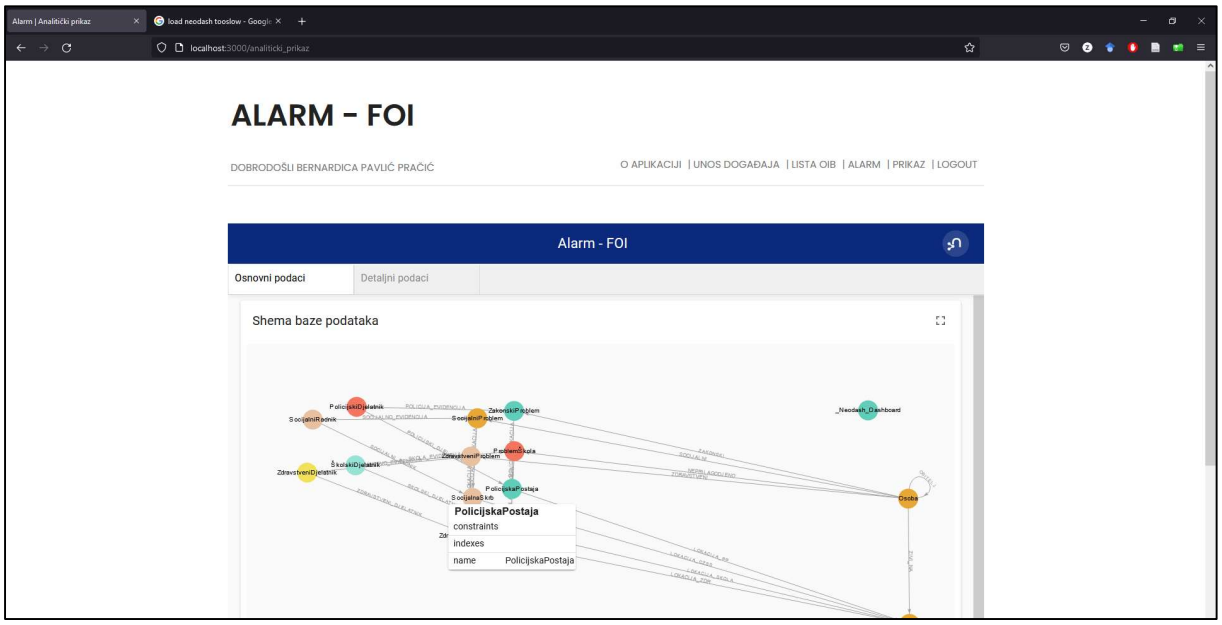
7.3. Vizualni prikaz

Kako bi osobe koje popunjavaju podatke imale osjećaj da se njihov trud isplati aplikacija osigurava dva prikaza. Prvi prikaz je tablica desetoro najugroženije djece koji ima u sebi gumb za slanje dodatnog nadzora za dijete. Nakon klika na gumb dijete se uklanja s popisa pošto je isto u programu praćenja pa nema potrebe za daljnje dodatno upozorenje.



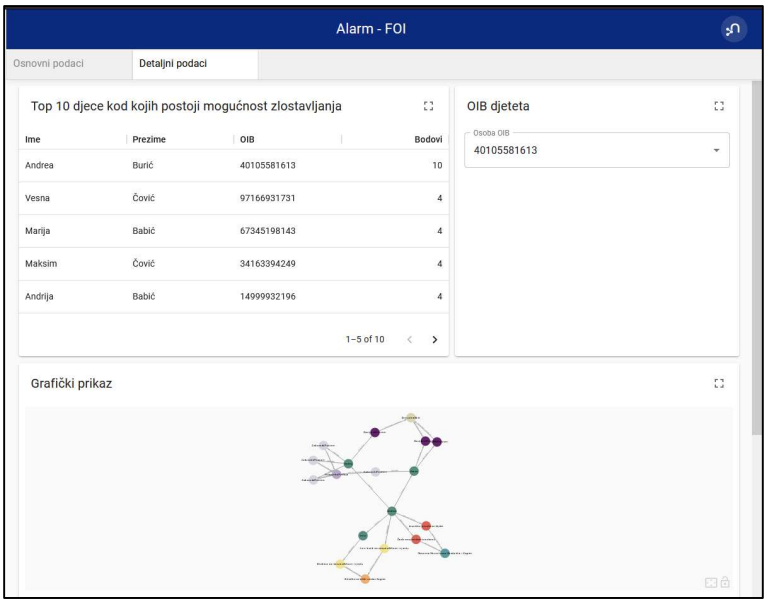
Slika 19 - Alarm prikaz desetoro najugroženije djece

Drugi prikaz je zanimljiviju pošto daje vizualni prikaz uz interaktivnost. Taj prikaz se zapravo opet dijeli na dva dijela. Na prvom dijelu je moguće vidjeti model baze podataka iz koje se da iščitati kakvi podaci su obuhvaćeni aplikacijom i bazom podataka. Iz njega je moguće lakše shvatiti koji podaci se unose u aplikaciju kako bi se dobila rang lista (Slika 20).



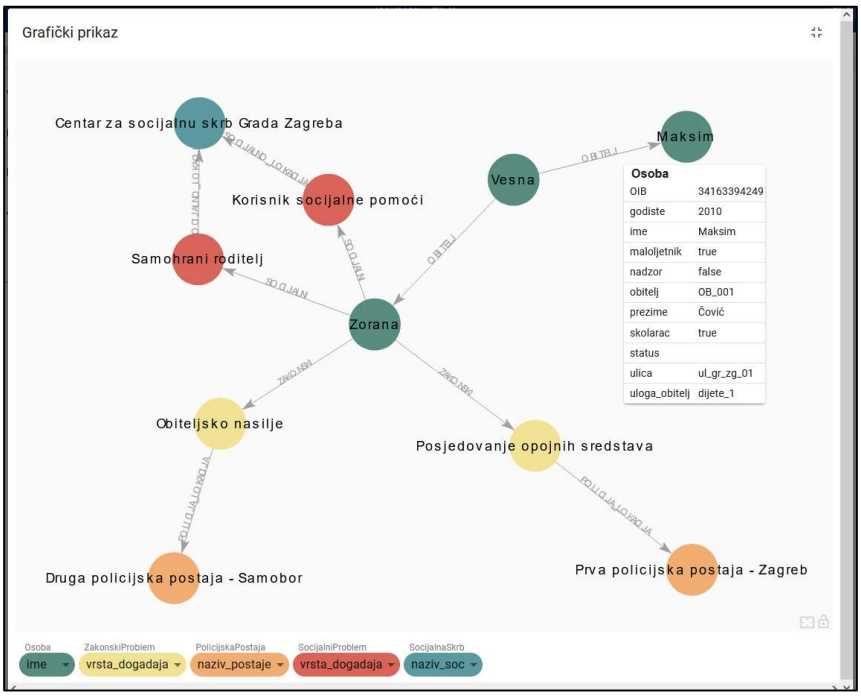
Slika 20 - Vizualni prikaz sheme

Drugi dio prikaza daje rang listu djece s OIB-om i brojem bodova. Osim toga služe tome da korisnik lakše dođe do OIB-a koji unosi u formu kako bi se prikazala ukupna situacija vezana uz dijete (Slika 21).



Slika 21 - Prikaz liste i grafički prikaz slučaja

Grafički prikaz nakon unosa OIB-a se nalazi ispod forme za unos. NeoDash omogućuje da se svaki dio prikaza uveća pa se tako može bolje vidjeti rezultat. Na Slici 22 je vidljivo kako se nakon klika mišem na čvor prikazuju svi njegovi podaci, a na dnu se biraju podaci koji će biti prikazani na čvorovima. NeoDash omogućuje da se isti odabiri i isključe, ali zbog preglednosti su navedene funkcije uključene u prikaz.



Slika 22 - Uvećan grafički prikaz slučaja

8. Zaključak

Jedan od ključnih elemenata koji se trebaju uzeti u obzir pri izradi aplikacije je svakako vremensko ograničenje koje razvojni tim ima pred sobom. Uvijek se s jedne strane nalaze mnoge želje naručitelja aplikacije, a s druge strane mogućnosti onoga koji aplikaciju razvija. Tek kada se te dvije strane usklade moguće je dobiti aplikaciju koja je kompromis dviju strana i s istom se može obaviti posao koji se od nje traži. U slučaju razvoja aplikacije za ovaj završni rad „naručitelj“ je bio Fakultet organizacije i informatike, a izvođač je student. Osim određenih funkcionalnosti aplikacije (mora raditi s Neo4j bazama podataka u pozadini) svakako treba obratiti pozornost na vrijeme dovršetka aplikacije. Autor ovog rada je krenuo sa savladavanjem samog sustava za upravljanje bazom podataka Neo4j, a nakon toga se trebalo odlučiti na kojem programskom jeziku će aplikacija biti sačinjena. Prva ideja je bila kako će se aplikacija sačiniti uz pomoć Angular okvira. Međutim, vrijeme koje je potrebno da bi se Angular savladao i s istim proizvela aplikacija je predugo, imajući u vidu vremensko ograničenje izrade rada. Nakon mjesec dana proučavanja Angulara, a približavanjem zadanih rokova, odluka je pala da se izabere neki drugi okvir. Druga opcija je bila PHP pošto se na smjeru Primjena informacijskih tehnologija u poslovanju tijekom šestog semestra taj jezik proučava u sklopu kolegija Izgradnja Web aplikacija. Međutim, kako Neo4j ne nudi službeno upravljački program za taj jezik, već iste razvijaju volonterski mnogi širom svijeta, odustalo se i od tog jezika. Na kraju je odabran Node.js. Osim što Neo4j nudi službeni upravljački program za taj okvir, učenje je kudikamo jednostavnije i brže. U pomoć svakako uskaču mnogi dodaci koje je razvila zajednica koja se okupila oko Node.js kroz platformu npm. Prvi koraci u Node.js su napravljeni početkom travnja da bi već krajem lipnja aplikacija bila u visokom stupnju razvoja i dokumentiranosti. To govori o brzini savladavanja osnovnih postulata razvoja aplikacije tim okvirom.

Kada se gleda završni proizvod, primjetno je kako je aplikacija jednostavna. To može biti prednost kada se ista želi čim prije implementirati. Edukacija korisnika bi se mogla svesti tek na kratku napisanu uputu za uporabu s par slika. Nema potrebe za dugačkim edukacijama i odsutnošću djelatnika s posla, a s druge strane aplikacija se može skoro odmah početi koristiti.

Tijekom razvoja aplikacije, a prije implementacije nekih sigurnosnih mehanizama, ista je bila postavljana na nekim online servisima za smještanje aplikacija. Heroku se pokazao kao poprilično ćudljiva platforma za smještaj aplikacije online u slučaju kada aplikacija leži na Node.js okviru. Zbog toga se aplikacija pokušala smjestiti na Azure platformu. Ta platforma je funkcionirala kako se i zamislilo, ali na žalost, nakon implementacije auth0 sustava autorizacije i autentifikacije dolazi do problema i s tom platformom. Zbog svega se odlučilo kako će

aplikacija biti samostalna na osobnom računalu korisnika. Postoji nekoliko sustava pakiranja Node.js aplikacije u izvršnu datoteku, a dodatak „pkg“ se odabrao zbog jednostavnosti implementacije. Prednost samostalne aplikacije na osobnom računalu je sigurnost. Puno manja je mogućnost napada na sustav pošto je isti distribuiran širom državnog aparata. U ovom slučaju jedino baza podataka leži na serveru koji se nalazi na Internetu.

Ono što bi se svakako moglo još bolje odraditi je korisničko sučelje. Da bi se podaci unijeli u bazu potrebno je znati OIB osobe za koju se podaci unose. Kontrola tog broja je odrađena na rudimentarnoj razini i to bi svakako bilo nešto što bi se moglo nadograditi kako bi korisnici imali bolji osjećaj tijekom korištenja aplikacije.

Sljedeće što bi se moglo poboljšati je svakako upravljanje pristupa pojedinim podacima osoba koje su u bazi podataka. Trebalo bi implementirati pravila GDPR (eng. *General Data Protection Regulation*) te odrediti što bi koji od korisnika aplikacije mogao vidjeti od osjetljivih podataka.

Mada aplikacija koja se nalazi na osobnom računalu korisnika ima određene prednosti, web aplikacija bi ipak bila jedan korak dalje. Pri tome bi trebalo riješiti dva problema. Prvi je već spomenut, a to je implementacija sustava autorizacije i autentifikacije. Drugi, veći problem bi tada bila sigurnost informacijskog sustava unutar organizacije koja koristi aplikaciju. Sve institucije predviđene ovim programom (policija, zdravstvo, školstvo i socijalna služba) rukuju s mnogim osjetljivim podacima. Aplikacija koja bi se nalazila na Internetu bi bila sigurnosni izazov za sve te institucije pošto bi jedan dio podataka mogao biti izložen. Da bi aplikacija postigla puni potencijal, pošto traži suradnju raznih institucija, trebala bi biti svima dostupna. Ali upravo to može biti problem pri planiranju i implementaciji aplikacije.

Još jedna stvar koja bi se svakako mogla poboljšati, ali zahtjeva nešto što nadilazi ovaj rad, je algoritam kojim se računa razina ugroženosti djeteta. Da bi se došlo do formule koja s velikom sigurnošću računa ugroženost djeteta potrebno je provesti veliko, sveobuhvatno multidisciplinarno i multiagencijsko istraživanje. Sadašnje statistike pokazuju tek parcijalne elemente te se iz njih ne može stvoriti jedna velika slika koja može dati podatak o, na primjer, tome koliko osoba koje narušavaju javni red i mir isto tako zlostavljaju svoje dijete. Ili na primjer ima li veze između zlostavljanja i obiteljskog nasilja i zlouporabe opojnih sredstava. Sva ta pitanja bi se mogla istražiti, ali to bi svakako bilo istraživanje na puno višoj razini i trebalo bi se angažirati mnoge institucije koje djeluju na području jedne države.

Popis literature

[1] M. Maleković i K. Rabuzin, *Uvod u baze podataka*. Varaždin: Fakultet organizacije i informatike, 2016.

[2] „The Convention on the Rights of the Child: The children's version". https://www.unicef.org/child-rights-convention/convention-text-childrens-version (pristupljeno 07. svibanj 2022.).

[3] „Violence against children". https://www.who.int/health-topics/violence-against-children (pristupljeno 23. lipanj 2022.).

[4] C. E. Cox, J. B. Kotch, i M. D. Everson, „A longitudinal study of modifying influences in the relationship between domestic violence and child maltreatment", *J. Fam. Violence*, sv. 18, izd. 1, str. 5–17, 2003.

[5] S. O'Connell, „Spotting the early signs of abuse: New draft guidance", *Br. J. Sch. Nurs.*, sv. 12, izd. 4, str. 187–189, 2017.

[6] K. Wells, „Substance abuse and child maltreatment", *Pediatr. Clin. North Am.*, sv. 56, izd. 2, str. 345–362, 2009.

[7] „Mali Lošinj subotu i nedjelju proglasio danima žalosti, zastave se spuštaju na pola koplja, a na javnim mjestima neće biti kulturnih ili zabavnih događaja", *Dnevnik.hr*. https://dnevnik.hr/vijesti/crna-kronika/u-malom-losinju-pronadjeno-tijelo-mladje-zene-policija-istrazuje-jednu-osobu---593941.html (pristupljeno 23. lipanj 2022.).

[8] „Sažetak izvješća o radu pravobraniteljice za djecu - 2019". Pristupljeno: 07. svibanj 2022. [Na internetu]. Dostupno na: https://dijete.hr/hr/download/sazetak-izvjesca-o-radu-pravobraniteljice-za-djecu-2019/?wpdmdl=14800&refresh=62764567af9d91651918183

[9] „Sažetak izvješća o radu pravobraniteljice za djecu - 2020". Pristupljeno: 07. svibanj 2022. [Na internetu]. Dostupno na: https://dijete.hr/hr/download/sazetak-izvjesca-o-radu-pravobraniteljice-za-djecu-za-2020-godinu/?wpdmdl=15780&refresh=627645a52d2471651918245

[10] „Sažetak izvješća o radu pravobraniteljice za djecu - 2021". Pristupljeno: 07. svibanj 2022. [Na internetu]. Dostupno na: https://dijete.hr/hr/download/sazetak-izvjesca-o-radu-pravobraniteljice-za-djecu-za-2021/?wpdmdl=16779&refresh=62764567a9a171651918183

[11] „Neo4j", *Megatrend*. https://www.megatrend.com/neo4j/ (pristupljeno 24. lipanj 2022.).

[12] G. Jordan, *Practical Neo4j*. Apress, 2014.

[13] „The Implementation Knowledge Graph of Air Crash Data based on Neo4j". http://ieeexplore.ieee.org/document/9085182/ (pristupljeno 24. lipanj 2022.).

[14] M. Needham i A. E. Hodler, „Graph Algorithms", str. 266.

[15] „mrežni graf | Hrvatska enciklopedija". https://www.enciklopedija.hr/natuknica.aspx?id=70130 (pristupljeno 24. lipanj 2022.).

[16] „dijagram | Hrvatska enciklopedija". https://www.enciklopedija.hr/Natuknica.aspx?ID=15088 (pristupljeno 24. lipanj 2022.).

[17] M. Maleković i M. Schatten, *Teorija i primjena baza podataka*. Varaždin: Fakultet organizacije i informatike, 2017.

[18] T. W. Ling i G. Dobbie, *Semistructured database design*, sv. 1. Springer Science & Business Media, 2004.

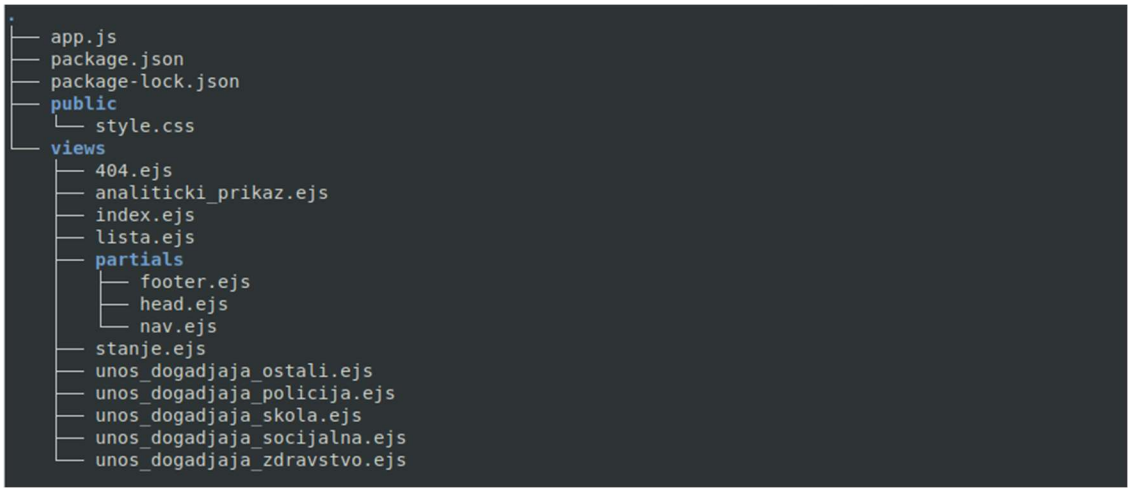
[19] T. King, „The 12 Best Graph Databases to Consider for 2022", *Best Data Management Software, Vendors and Data Science Platforms*, 22. prosinac 2021.

https://solutionsreview.com/data-management/the-best-graph-databases/ (pristupljeno 25. lipanj 2022.).
[20] B. M. Sasaki, „RDBMS & Graphs: SQL vs. Cypher Query Languages“, *Neo4j Graph Data Platform*, 07. ožujak 2016. https://neo4j.com/blog/sql-vs-cypher-query-languages/ (pristupljeno 15. srpanj 2022.).
[21] „System requirements - Operations Manual“, *Neo4j Graph Data Platform*. https://neo4j.com/docs/operations-manual/4.4/installation/requirements/ (pristupljeno 25. lipanj 2022.).
[22] „Product“, *Neo4j Graph Data Platform*. https://neo4j.com/product/ (pristupljeno 26. lipanj 2022.).
[23] „Comparing SQL with Cypher - Developer Guides“, *Neo4j Graph Data Platform*. https://neo4j.com/developer/cypher/guide-sql-to-cypher/ (pristupljeno 26. lipanj 2022.).
[24] O. Panzarino, *Learning cypher: write powerful and efficient queries for Neo4j with Cypher its official query language*. Birmingham, U.K: Packt Pub, 2014.
[25] „MATCH - Neo4j Cypher Manual“, *Neo4j Graph Data Platform*. https://neo4j.com/docs/cypher-manual/4.4/clauses/match/ (pristupljeno 27. lipanj 2022.).
[26] „SET - Neo4j Cypher Manual“, *Neo4j Graph Data Platform*. https://neo4j.com/docs/cypher-manual/4.4/clauses/set/ (pristupljeno 27. lipanj 2022.).
[27] „Graph database concepts - Getting Started“, *Neo4j Graph Data Platform*. https://neo4j.com/docs/getting-started/4.4/graphdb-concepts/ (pristupljeno 29. lipanj 2022.).
[28] „Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel“. https://www.mockaroo.com/ (pristupljeno 27. lipanj 2022.).
[29] „Značenje imena“. https://www.znacenje-imena.com/ (pristupljeno 27. lipanj 2022.).
[30] A. Sweigart, *Automate the boring stuff with Python: practical programming for total beginners*. No Starch Press, 2019.
[31] „CSV file: Definition - Google Ads Help“. https://support.google.com/google-ads/answer/9004364?hl=en (pristupljeno 27. lipanj 2022.).
[32] „Provjerite popis od 33.000 prezimena i saznajte koliko ih u Hrvatskoj nosi vaše“, *mirovina.hr*, 29. listopad 2018. https://www.mirovina.hr/novosti/provjerite-popis-33-000-prezimena-saznajte-koliko-ih-hrvatskoj-nosi-vase/ (pristupljeno 27. lipanj 2022.).
[33] „Importing CSV Data into Neo4j - Developer Guides“, *Neo4j Graph Data Platform*. https://neo4j.com/developer/guide-import-csv/ (pristupljeno 29. lipanj 2022.).
[34] „Constraints - Neo4j Cypher Manual“, *Neo4j Graph Data Platform*. https://neo4j.com/docs/cypher-manual/4.4/constraints/ (pristupljeno 30. lipanj 2022.).
[35] „MERGE - Neo4j Cypher Manual“, *Neo4j Graph Data Platform*. https://neo4j.com/docs/cypher-manual/4.4/clauses/merge/ (pristupljeno 29. lipanj 2022.).
[36] B. M. Sasaki, „Graph Visualization for Neo4j Schemas using yFiles“, *Neo4j Graph Data Platform*, 04. svibanj 2017. https://neo4j.com/blog/graph-visualization-neo4j-schemas-yfiles/ (pristupljeno 30. lipanj 2022.).
[37] „DELETE - Neo4j Cypher Manual“, *Neo4j Graph Data Platform*. https://neo4j.com/docs/cypher-manual/4.4/clauses/delete/ (pristupljeno 30. lipanj 2022.).
[38] „Introduction to Node.js“, *Introduction to Node.js*. https://nodejs.dev/learn (pristupljeno 30. lipanj 2022.).
[39] „Differences between Node.js and the Browser“, *Differences between Node.js and the Browser*. https://nodejs.dev/learn/differences-between-nodejs-and-the-browser (pristupljeno 30. lipanj 2022.).
[40] „Node.js“. https://nodejs.org/en/ (pristupljeno 30. lipanj 2022.).

[41] „npm About“. https://www.npmjs.com/about (pristupljeno 30. lipanj 2022.).
[42] „Create Your First Node js app with Express“, *JSFORALL*, 29. prosinac 2019. https://jsforall.com/nodejs/steps-to-create-first-nodejs-express-app/ (pristupljeno 30. lipanj 2022.).
[43] „express“, *npm*. https://www.npmjs.com/package/express (pristupljeno 30. lipanj 2022.).
[44] „ejs“, *npm*. https://www.npmjs.com/package/ejs (pristupljeno 14. svibanj 2022.).
[45] „express-openid-connect“, *npm*. https://www.npmjs.com/package/express-openid-connect (pristupljeno 01. srpanj 2022.).
[46] „neo4j-driver“, *npm*. https://www.npmjs.com/package/neo4j-driver (pristupljeno 14. svibanj 2022.).
[47] „nodemon“, *npm*. https://www.npmjs.com/package/nodemon (pristupljeno 30. lipanj 2022.).
[48] „opener“, *npm*. https://www.npmjs.com/package/opener (pristupljeno 21. srpanj 2022.).
[49] „Express basic routing“. https://expressjs.com/en/starter/basic-routing.html (pristupljeno 02. srpanj 2022.).
[50] *Authentication vs Authorization | Identity 101*, (2022.). Pristupljeno: 02. srpanj 2022. [Na internetu Video]. Dostupno na: https://www.youtube.com/watch?v=2iTTBJd46ow
[51] Auth0, „Auth0 Overview“, *Auth0 Docs*. https://auth0.com/docs/ (pristupljeno 02. srpanj 2022.).
[52] R. Brath, *Graph analysis and visualization: discovering business opportunity in linked data*. Indianapolis, IN: Wiley, 2015.
[53] „NeoDash - Neo4j Dashboard Builder“. http://neodash.graphapp.io/ (pristupljeno 14. srpanj 2022.).
[54] „pkg“, *npm*. https://www.npmjs.com/package/pkg (pristupljeno 12. srpanj 2022.).

Popis slika

Prilog – programski kod



Slika 23 – Prikaz stabla mapa i datoteka aplikacije

/app.js

```
const express = require('express');
const neo4j = require('neo4j-driver');
const bodyParser = require('body-parser');
const { auth } = require('express-openid-connect');

const opener = require('opener');
opener('http://localhost:3000');

const config = {
  authRequired: false,
  auth0Logout: true,
  secret:
"lčklkjslkdfnvpoisevosidjgćlskvdćydjvčlkevnćvmčylvčvlčmsčlcvsdmvčdfkjv",
  baseURL: "http://localhost:3000",
  clientID: "YvnwhhthjSBkMGY4dJNcbZLWLJAz1R9T",
  issuerBaseURL: "https://dev-xhzjybdg.us.auth0.com",
};
const app = express();

app.set('view engine', 'ejs');
app.use(express.static ('public'));
app.use(express.static ('views'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: false}));
app.use(auth(config));
```

```
const port = 3000;
app.listen(port);

module.exports = app;

// Otvaranje sesije
var driver = neo4j.driver('neo4j+s://36e5e105.databases.neo4j.io',
neo4j.auth.basic('neo4j', 'Eg7O6ak1ErjFPeCwsi-IMcQds9gWKcQDDUx8jBN9mco'));
session = driver.session();


// Popis osoba
app.get('/stanje', function(req, res){
  session
    .run('MATCH (n:Osoba) RETURN n')
    .then(function(result){
        var osoba = [];
        result.records.forEach(function(record){
            osoba.push({
                id: record._fields[0].identity.low,
                ime: record._fields[0].properties.ime,
                prezime: record._fields[0].properties.prezime,
                godiste: record._fields[0].properties.godiste,
                OIB: record._fields[0].properties.OIB
            });
        });
        res.render('stanje', { title: 'Lista',
            osobe: osoba,
            isAuthenticated: req.oidc.isAuthenticated(),
            user: req.oidc.user,
        });
        session.close;
    })
    .catch(function(err){
        console.log(err);
    });
});

// Lista ALARM
app.get('/lista', function(req, res){
  session
    .run
'MATCH (o1:Osoba)-[r1:OBITELJ]->(o2:Osoba)
WHERE (date().year - toInteger(o1.godiste)) < 18 AND o1.nadzor = "false"
OPTIONAL MATCH (o1)-[r1]->(ps:`ProblemŠkola`)
OPTIONAL MATCH (o1)-[r2]->(zdp:`ZdravstveniProblem`)
OPTIONAL MATCH (o1)-[r3]->(o2:Osoba)-[r4]->(zp:ZakonskiProblem)
OPTIONAL MATCH (o1)-[r5]->(o2:Osoba)-[r6]->(sp:SocijalniProblem)
```

```
RETURN o1, count(distinct r1)+count(distinct r2)+count(distinct
r4)+count(distinct r6) AS Bodovi
ORDER BY Bodovi
DESC LIMIT 10')
    .then(function(result){
        var osoba = [];
        result.records.forEach(function(record){
            osoba.push({
                ime: record._fields[0].properties.ime,
                prezime: record._fields[0].properties.prezime,
                OIB: record._fields[0].properties.OIB,
                nadzor: record._fields[0].properties.nadzor
            });
        });
        res.render('lista', { title: 'ALARM',
            osobe: osoba,
            isAuthenticated: req.oidc.isAuthenticated(),
            user: req.oidc.user,
        });
        session.close;
    })
    .catch(function(err){
        console.log(err);
    });
});

// Poslan nadzor
app.post('/lista', function(req, res){

  var nadzor_OIB = req.body.nadzor_OIB;

      session
      .run('MATCH (o:Osoba{OIB: $nadzor_OIB}) SET o.nadzor = "true" RETURN
o', {nadzor_OIB: nadzor_OIB})
      .then(function(result){
        res.redirect('/lista');
        session.close;
        })
      .catch(function(err){
        console.log(err);
        });
      });

// Dodavanje događaja - Policija
app.post('/dogadjaj/dodaj_policija', function(req, res){

  var dog_OIB = req.body.dog_OIB;
  var id_ustanova = req.body.id_ustanova;
  var vrijeme = req.body.dog_vrijeme;
```

```
  var vrsta_dogadjaj = req.body.vrsta_dogadjaj;

      session
      .run('MATCH (o:Osoba{OIB: $dog_OIB}),
(d:PolicijskiDjelatnik{id_djelatnik: "pol_001"}),
(i:PolicijskaPostaja{id_ustanova: $id_ustanova}) MERGE (o)-[r1:ZAKONSKI]-
>(p:ZakonskiProblem{vrijeme: $vrijeme, vrsta_dogadaja: $vrsta_dogadjaj})
MERGE (d)-[r2:POLICIJA_EVIDENCIJA]->(p) MERGE (p)-[r3:POLICIJA_LOKACIJA]-
>(i) RETURN o, d, i, p', {dog_OIB: dog_OIB, id_ustanova: id_ustanova,
vrijeme: vrijeme, vrsta_dogadjaj: vrsta_dogadjaj, })
      .then(function(result){
        res.redirect('/unos_dogadjaja_policija');
        session.close;
        })
      .catch(function(err){
        console.log(err);
        });
      });

// Dodavanje događaja - Škola
app.post('/dogadjaj/dodaj_skola', function(req, res){

  var dog_OIB = req.body.dog_OIB;
  var id_ustanova = req.body.id_ustanova;
  var vrijeme = req.body.dog_vrijeme;
  var vrsta_dogadjaj = req.body.vrsta_dogadjaj;

      session
      .run('MATCH (o:Osoba{OIB: $dog_OIB}),
(d:ŠkolskiDjelatnik{id_djelatnik: "skol_001"}),
(i:ŠkolskaUstanova{id_ustanova: $id_ustanova}) MERGE (o)-
[r1:NEPRILAGODJENO]->(p:ProblemŠkola{vrijeme: $vrijeme, vrsta_dogadaja:
$vrsta_dogadjaj}) MERGE (d)-[r2:SKOLA_EVIDENCIJA]->(p) MERGE (p)-
[r3:SKOLA_LOKACIJA]->(i) RETURN o, d, i, p', {dog_OIB: dog_OIB,
id_ustanova: id_ustanova, vrijeme: vrijeme, vrsta_dogadjaj: vrsta_dogadjaj,
})
      .then(function(result){
        res.redirect('/unos_dogadjaja_skola');
        session.close;
        })
      .catch(function(err){
        console.log(err);
        });
      });

// Dodavanje događaja - Zdravstvo
app.post('/dogadjaj/dodaj_zdravstvo', function(req, res){

  var dog_OIB = req.body.dog_OIB;
```

```
  var id_ustanova = req.body.id_ustanova;
  var vrijeme = req.body.dog_vrijeme;
  var vrsta_dogadjaj = req.body.vrsta_dogadjaj;

      session
      .run('MATCH (o:Osoba{OIB: $dog_OIB}),
(d:ZdravstveniDjelatnik{id_djelatnik: "zdr_001"}),
(i:ZdravstvenaUstanova{id_ustanova: $id_ustanova}) MERGE (o)-
[r1:ZDRAVSTVENI]->(p:ZdravstveniProblem{vrijeme: $vrijeme, vrsta_dogadaja:
$vrsta_dogadjaj}) MERGE (d)-[r2:ZDRAVSTVO_EVIDENCIJA]->(p) MERGE (p)-
[r3:ZDRAVSTVO_LOKACIJA]->(i) RETURN o, d, i, p', {dog_OIB: dog_OIB,
id_ustanova: id_ustanova, vrijeme: vrijeme, vrsta_dogadjaj: vrsta_dogadjaj,
})
      .then(function(result){
        res.redirect('/unos_dogadjaja_zdravstvo');
        session.close;
        })
      .catch(function(err){
        console.log(err);
        });
      });


// Dodavanje događaja - Socijalna služba
app.post('/dogadjaj/dodaj_socijalna', function(req, res){

  var dog_OIB = req.body.dog_OIB;
  var id_ustanova = req.body.id_ustanova;
  var vrijeme = req.body.dog_vrijeme;
  var vrsta_dogadjaj = req.body.vrsta_dogadjaj;

      session
      .run('MATCH (o:Osoba{OIB: $dog_OIB}),
(d:SocijalniRadnik{id_djelatnik: "ss_001"}), (i:SocijalnaSkrb{id_ustanova:
$id_ustanova}) MERGE (o)-[r1:SOCIJALNI]->(p:SocijalniProblem{vrijeme:
$vrijeme, vrsta_dogadaja: $vrsta_dogadjaj}) MERGE (d)-
[r2:SOCIJALNO_EVIDENCIJA]->(p) MERGE (p)-[r3:SOCIJALNO_LOKACIJA]->(i)
RETURN o, d, i, p', {dog_OIB: dog_OIB, id_ustanova: id_ustanova, vrijeme:
vrijeme, vrsta_dogadjaj: vrsta_dogadjaj, })
      .then(function(result){
        res.redirect('/unos_dogadjaja_socijalna');
        session.close;
        })
      .catch(function(err){
        console.log(err);
        });
      });
```

```
// Routes
app.get('/', (req, res) => {
  res.render('index', { title: 'O aplikaciji',
  isAuthenticated: req.oidc.isAuthenticated(),
  user: req.oidc.user,
});
});

app.get('/unos_dogadjaja_policija', (req, res) => {
  res.render('unos_dogadjaja_policija', { title: 'Unos događaja -
Policija',
  isAuthenticated: req.oidc.isAuthenticated(),
  user: req.oidc.user
 });
});

app.get('/unos_dogadjaja_skola', (req, res) => {
  res.render('unos_dogadjaja_skola', { title: 'Unos događaja - Škola',
  isAuthenticated: req.oidc.isAuthenticated(),
  user: req.oidc.user
 });
});

app.get('/unos_dogadjaja_socijalna', (req, res) => {
  res.render('unos_dogadjaja_socijalna', { title: 'Unos događaja -
Socijalna služba',
  isAuthenticated: req.oidc.isAuthenticated(),
  user: req.oidc.user
 });
});

app.get('/unos_dogadjaja_zdravstvo', (req, res) => {
  res.render('unos_dogadjaja_zdravstvo', { title: 'Unos događaja -
Zdravstvo',
  isAuthenticated: req.oidc.isAuthenticated(),
  user: req.oidc.user
 });
});

app.get('/unos_dogadjaja_ostali', (req, res) => {
  res.render('unos_dogadjaja_ostali', { title: 'Unos događaja - Ostali',
  isAuthenticated: req.oidc.isAuthenticated(),
  user: req.oidc.user
 });
});

app.use((req, res) => {
  res.status(404).render('404', { title: 'Nepostojeća stranica',
  isAuthenticated: req.oidc.isAuthenticated(),
```

```
  user: req.oidc.user,
});
});
```

/package.json

```
{
  "name": "alarm",
  "version": "1.0.0",
  "engines": {
    "node": "16.x",
    "npm": "8.x"
  },
  "description": "",
  "main": "app.js",
  "bin": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node app.js"
  },
  "keywords": [
    "alarm",
    "zlostavljanje",
    "dijete",
    "završni",
    "rad",
    "PITUP",
    "FOI",
    "Varaždin"
  ],
  "author": "Zlatko Pračić",
  "license": "ISC",
  "dependencies": {
    "dotenv": "^16.0.1",
    "ejs": "^3.1.8",
    "express": "^4.18.1",
    "express-openid-connect": "^2.7.2",
    "neo4j-driver": "*",
    "nodemon": "^1.3.3",
    "opener": "^1.5.2"
  },
  "pkg": {
    "assets": [
      "views/**/*",
      "public/**/*",
      "node_modules/**/*"
    ]
  }
```

```
}
```

/public/style.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=
swap');
body {
  max-width: 1200px;
  margin: 20px auto;
  padding: 0 20px;
  font-family: 'Poppins', sans-serif;;
  max-width: 1200px;
}
p,
h1,
h2,
h3,
a,
ul,
td {
  margin: 0;
  padding: 5;
  text-decoration: none;
  color: #222;
}

/* nav & footer styles */
nav {
  display: flex;
  justify-content: space-between;
  margin-bottom: 60px;
  padding-bottom: 10px;
  border-bottom: 1px solid #ddd;
  text-transform: uppercase;
}
nav ul {
  display: flex;
  justify-content: space-between;
  align-items: flex-end;
}
nav li {
  list-style-type: none;
  margin-left: 20px;
}
nav h1 {
  font-size: 3em;
```

```
}
nav p,
nav a {
  color: #777;
  font-weight: 300;
}
footer {
  color: #777;
  text-align: center;
  margin: 80px auto 20px;
}
h2 {
  margin-bottom: 40px;
}
h3 {
  text-transform: capitalize;
  margin-bottom: 8px;
}
.content {
  margin-left: 20px;
}

form {
  max-width: 400px;
  margin: 0 auto;
}
input,
textarea {
  display: block;
  width: 100%;

  margin: 10px 0;
  padding: 8px;
}
label {
  display: block;
  margin-top: 24px;
}
textarea {
  height: 120px;
}
button {
  margin-top: 3px;
  margin-bottom: 3px;
  background: #025E73;
  color: white;
  padding: 6px;
  border: 0;
  font-size: 1.2em;
```

```
  cursor: pointer;
}

button:hover {
  background: #17aecf;
}

select {
  vertical-align: middle;
  margin: 5px 10px;
  padding: 10px;
  color: #fff;
  background-color: #025E73;
  border: 1px solid #ddd;
}
```

/views/404.ejs

```
<html lang="hr">
  <%- include("./partials/head.ejs") %>
  <body>
    <%- include("./partials/nav.ejs") %>
    <h1>404!</h1>
    <h2>Ta stranica ne postoji</h2>
    <%- include("./partials/footer.ejs") %>
  </body>
</html>
```

/views/index.ejs

```
<html lang="hr">
  <%- include("./partials/head.ejs") %>
  <body>
    <%- include("./partials/nav.ejs") %>
    <h1>O aplikaciji</h1>
    <h2>nešto o aplikaciji</h2>

    <%- include("./partials/footer.ejs") %>
  </body>
</html>
```

/views/lista.ejs

```
<html lang="hr">
  <%- include("./partials/head.ejs") %>
  <body>
```

```
    <%- include("./partials/nav.ejs") %>

    <h1>Alarm</h1>

    <table id="myTable" border="1" style="width: 99%">
      <thead>
        <tr>
          <th style="width: 20%">Ime</th>
          <th style="width: 20%">Prezime</th>
          <th style="width: 20%">OIB</th>
          <th style="width: 20%">Nadzor</th>
        </tr>
      </thead>
      <tbody>
        <% osobe.forEach(function(osoba){ %> <% if (osoba.nadzor ==
'false') {
        %>
        <tr>
          <td><%= osoba.ime %></td>
          <td><%= osoba.prezime %></td>
          <td><%= osoba.OIB %></td>
          <td style="text-align: center">
            <form action="lista" method="post">
              <input type="hidden" name="nadzor_OIB" value="<%= osoba.OIB
%>" />
              <button type="submit">Poslati nadzor</button>
            </form>
          </td>
        </tr>
        <% } %> <% }) %>
      </tbody>
    </table>
    <%- include("./partials/footer.ejs") %>
  </body>
</html>
```

/views/stanje.ejs

```
<html lang="hr">
  <%- include("./partials/head.ejs") %>
  <body>
    <%- include("./partials/nav.ejs") %>

    <h1>Provjera osobe</h1>

    <table style="width: 99%">
      <tr>
        <td style="width: 25%">
```

```
        <label for="myInput1" name="myInput1">Ime: </label>
        <input
          type="text"
          id="myInput1"
          onkeyup="myFunctionIme()"
          placeholder="Pretraga po imenu.."
          title="Upiši ime"
        />
      </td>
      <td style="width: 25%">
        <label for="myInput2" name="myInput2">Prezime: </label>
        <input
          type="text"
          id="myInput2"
          onkeyup="myFunctionPrezime()"
          placeholder="Pretraga po prezimenu.."
          title="Upiši prezime"
        />
      </td>
      <td style="width: 25%">
        <label for="myInput3" name="myInput3">Godište: </label>
        <input
          type="text"
          id="myInput3"
          onkeyup="myFunctionGodiste()"
          placeholder="Pretraga po godištu.."
          title="Upiši godište"
        />
      </td>
      <td style="width: 25%">
        <label for="myInput4" name="myInput4">OIB: </label>
        <input
          type="text"
          id="myInput4"
          onkeyup="myFunctionOIB()"
          placeholder="Pretraga po OIB-u.."
          title="Upiši OIB"
        />
      </td>
    </tr>
  </table>

  <table id="myTable" border="1" style="width: 99%">
    <thead>
      <tr>
        <th style="width: 25%">Ime</th>
        <th style="width: 25%">Prezime</th>
        <th style="width: 25%">Godište</th>
        <th style="width: 25%">OIB</th>
```

```
      </tr>
    </thead>
    <tbody>
      <% osobe.forEach(function(osoba){ %>
      <tr>
        <td><%= osoba.ime %></td>
        <td><%= osoba.prezime %></td>
        <td><%= osoba.godiste %></td>
        <td><%= osoba.OIB %></td>
      </tr>
      <% }) %>
    </tbody>
  </table>

  <script>
    function myFunctionIme() {
      var input, filter, table, tr, td, i, txtValue;
      input = document.getElementById("myInput1");
      filter = input.value.toUpperCase();
      table = document.getElementById("myTable");
      tr = table.getElementsByTagName("tr");
      for (i = 0; i < tr.length; i++) {
        td = tr[i].getElementsByTagName("td")[0];
        if (td) {
          txtValue = td.textContent || td.innerText;
          if (txtValue.toUpperCase().indexOf(filter) > -1) {
            tr[i].style.display = "";
          } else {
            tr[i].style.display = "none";
          }
        }
      }
    }
  </script>

  <script>
    function myFunctionPrezime() {
      var input, filter, table, tr, td, i, txtValue;
      input = document.getElementById("myInput2");
      filter = input.value.toUpperCase();
      table = document.getElementById("myTable");
      tr = table.getElementsByTagName("tr");
      for (i = 0; i < tr.length; i++) {
        td = tr[i].getElementsByTagName("td")[1];
        if (td) {
          txtValue = td.textContent || td.innerText;
          if (txtValue.toUpperCase().indexOf(filter) > -1) {
            tr[i].style.display = "";
          } else {
```

```
            tr[i].style.display = "none";
          }
        }
      }
    }
  </script>

  <script>
    function myFunctionGodiste() {
      var input, filter, table, tr, td, i, txtValue;
      input = document.getElementById("myInput3");
      filter = input.value.toUpperCase();
      table = document.getElementById("myTable");
      tr = table.getElementsByTagName("tr");
      for (i = 0; i < tr.length; i++) {
        td = tr[i].getElementsByTagName("td")[2];
        if (td) {
          txtValue = td.textContent || td.innerText;
          if (txtValue.toUpperCase().indexOf(filter) > -1) {
            tr[i].style.display = "";
          } else {
            tr[i].style.display = "none";
          }
        }
      }
    }
  </script>

  <script>
    function myFunctionOIB() {
      var input, filter, table, tr, td, i, txtValue;
      input = document.getElementById("myInput4");
      filter = input.value.toUpperCase();
      table = document.getElementById("myTable");
      tr = table.getElementsByTagName("tr");
      for (i = 0; i < tr.length; i++) {
        td = tr[i].getElementsByTagName("td")[3];
        if (td) {
          txtValue = td.textContent || td.innerText;
          if (txtValue.toUpperCase().indexOf(filter) > -1) {
            tr[i].style.display = "";
          } else {
            tr[i].style.display = "none";
          }
        }
      }
    }
  </script>
```

```
    <%- include("./partials/footer.ejs") %>
  </body>
</html>
```

/views/analiticki_prikaz.ejs

```
<html lang="hr">
  <%- include("./partials/head.ejs") %>
  <body>
    <%- include("./partials/nav.ejs") %>
    <h1>O aplikaciji</h1>
    <h2>nešto o aplikaciji</h2>

    <%- include("./partials/footer.ejs") %>
  </body>
</html>
```

/views/partials/footer.ejs

```
<footer>
  Copyright &copy; Zlatko Pračić
  <a href="mailto:zpracic@foi.hr">zpracic@foi.hr</a> <%= new
  Date().getFullYear();%>.
</footer>
```

/views/partials/head.ejs

```
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Alarm | <%= title %></title>
  <link rel="stylesheet" href="/style.css" />
</head>
```

/views/partials/nav.ejs

```
<nav>
  <div class="site-title">
    <a href="/"><h1>Alarm - FOI</h1></a>
    <% if(user) { %>
    <p>Dobrodošli <%=user.name %></p>
    <% } else { %>
    <p>Neregistrirani korisnik</p>
    <% } %>
  </div>
```

```
  <ul>
    <a href="/">O aplikaciji</a>
    <% if(user) { %> <% if (user.email.split('@').reverse()[0] ===
"mup.hr") {
    %>
    <a href="/unos_dogadjaja_policija"> | Unos događaja - Policija</a>
    <% } else if (user.email.split('@').reverse()[0] === "czss.hr") { %>
    <a href="/unos_dogadjaja_socijalna"> | Unos događaja - CZSS</a>
    <% } else if (user.email.split('@').reverse()[0] === "miz.hr") { %>
    <a href="/unos_dogadjaja_zdravstvo"> | Unos događaja - Zdravstvo</a>
    <% } else if (user.email.split('@').reverse()[0] === "skole.hr") { %>
    <a href="/unos_dogadjaja_skola"> | Unos događaja - Škola</a>
    <% } else { %>
    <a href="/unos_dogadjaja_ostali"> | Unos događaja - Ostali</a>
    <% } %> <% } %> <% if(user) { %>
    <a href="/stanje"> | Ukupna lista</a>
    <a href="/lista"> | Alarm lista</a>
    <% } %> <% if(!isAuthenticated) { %>
    <a href="/login/"> | Login</a>
    <% } else { %>
    <a href="/logout/"> | Logout</a>
    <% } %>
  </ul>
</nav>
```

/views/unos_dogadjaja_ostali.ejs

```
<html lang="hr">
  <%- include("./partials/head.ejs") %>
  <body>
    <%- include("./partials/nav.ejs") %>
    <div>
      <h1>Nemate pravo unosa u aplikaciju</h1>
      <br />
    </div>
    <%- include("./partials/footer.ejs") %>
  </body>
</html>
```

/views/unos_dogadjaja_policija.ejs

```
<html lang="hr">

    <% if (user.email.split('@').reverse()[0] !== "mup.hr") { %>
        <script>window.location.href = "/";</script>
    <% } %>
```

```
    <%- include("./partials/head.ejs") %>
<body>
    <%- include("./partials/nav.ejs") %>
    <div>
        <h1>Dodaj događaj</h1><br>

            <form action="/dogadjaj/dodaj_policija" method="post">
            <label for="dog_OIB">OIB osobe:</label>
            <input type="text" name="dog_OIB" size="50"
required="required"><br>
            <label for="dog_vrijeme">Vrijeme događaja:</label>
            <input type="date" name="dog_vrijeme" required="required"><br>
            <label for="id_ustanova">Ustanova: </label>
                <select name="id_ustanova">
                    <option value="zg_1pp">1 (prva) policijska postaja -
Zagreb</option>
                    <option value="sam_2pp">2 (druga) policijska postaja -
Samobor</option>
                    <option value="ds_3pp">3 (treća) policijska postaja -
Dugo Selo</option>
                </select><br>
            <label for="vrsta_dogadjaj">Vrsta događaja: </label>
                <select name="vrsta_dogadjaj">
                    <option value="Posjedovanje opojnih
sredstava">Posjedovanje opojnih sredstava</option>
                    <option value="Obiteljsko nasilje">Obiteljsko
nasilje</option>
                </select><br><br>
                <button type="submit">Pošalji</button>
    </div>
    <%- include("./partials/footer.ejs") %>
</body>
</html>
```

/views/unos_dogadjaja_skola.ejs

```
<html lang="hr">

    <% if (user.email.split('@').reverse()[0] !== "skole.hr") { %>
        <script>window.location.href = "/";</script>
    <% } %>

<%- include("./partials/head.ejs") %>
<body>
    <%- include("./partials/nav.ejs") %>
    <div>
        <h1>Dodaj događaj</h1><br>
```

```
            <form action="/dogadjaj/dodaj_skola" method="post">
            <label for="dog_OIB">OIB osobe:</label>
            <input type="text" name="dog_OIB" size="50"
required="required"><br>
            <label for="dog_vrijeme">Vrijeme događaja:</label>
            <input type="date" name="dog_vrijeme" required="required"><br>
            <label for="id_ustanova">Ustanova: </label>
                <select name="id_ustanova">
                    <option value="zg_osig">Osnovna škola Ivana Gundulića -
Zagreb</option>
                    <option value="zg_osml">Osnovna škola Matka Laginje -
Zagreb</option>
                    <option value="sam_osbt">Osnovna škola Bogumila Tonija
- Samobor</option>
                    <option value="sam_osr">Osnovna škola Rude -
Samobor</option>
                    <option value="ds_osjz">Osnovna škola Josipa Zorića -
Dugo Selo</option>
                    <option value="ds_osib">Osnovna škola Ivan Benković -
Dugo Selo</option>
                </select><br>
            <label for="vrsta_dogadjaj">Vrsta događaja: </label>
                <select name="vrsta_dogadjaj">
                    <option value="Izuzetno povučeno dijete">Izuzetno
povučeno dijete</option>
                    <option value="Česti neopravdani izostanci">Česti
neopravdani izostanci</option>
                </select><br><br>
                <button type="submit">Pošalji</button>
    </div>
    <%- include("./partials/footer.ejs") %>
</body>
</html>
```

/views/unos_dogadjaja_socijalna.ejs

```
<html lang="hr">

    <% if (user.email.split('@').reverse()[0] !== "czss.hr") { %>
        <script>window.location.href = "/";</script>
    <% } %>

<%- include("./partials/head.ejs") %>
<body>
    <%- include("./partials/nav.ejs") %>
    <div>
        <h1>Dodaj događaj</h1><br>
```

```
            <form action="/dogadjaj/dodaj_socijalna" method="post">
            <label for="dog_OIB">OIB osobe:</label>
            <input type="text" name="dog_OIB" size="50"
required="required"><br>
            <label for="dog_vrijeme">Vrijeme događaja:</label>
            <input type="date" name="dog_vrijeme" required="required"><br>
            <label for="id_ustanova">Ustanova: </label>
                <select name="id_ustanova">
                    <option value="zg_czss">Centar za socijalnu skrb Grada
Zagreba</option>
                    <option value="sam_czss">Centar za socijalnu skrb Grada
Samobora</option>
                    <option value="ds_czss">Centar za socijalnu skrb Grada
Dugo Selo</option>
                </select><br>
            <label for="vrsta_dogadjaj">Vrsta događaja: </label>
                <select name="vrsta_dogadjaj">
                    <option value="Samohrani roditelj">Samohrani
roditelj</option>
                    <option value="Korisnik socijalne pomoći">Korisnik
socijalne pomoći</option>
                </select><br><br>
            <button type="submit">Pošalji</button>
    </div>
    <%- include("./partials/footer.ejs") %>
</body>
</html>
```

/views/unos_dogadjaja_zdravstvo.ejs

```
<html lang="hr">

    <% if (user.email.split('@').reverse()[0] !== "miz.hr") { %>
        <script>window.location.href = "/";</script>
    <% } %>

<%- include("./partials/head.ejs") %>
<body>
    <%- include("./partials/nav.ejs") %>
    <div>
        <h1>Dodaj događaj</h1><br>

            <form action="/dogadjaj/dodaj_zdravstvo" method="post">
            <label for="dog_OIB">OIB osobe:</label>
            <input type="text" name="dog_OIB" size="50"
required="required"><br>
            <label for="dog_vrijeme">Vrijeme događaja:</label>
            <input type="date" name="dog_vrijeme" required="required"><br>
```

```
            <label for="id_ustanova">Ustanova: </label>
                <select name="id_ustanova">
                    <option value="zg_KBC">Klinički bolnički centar - 
Zagreba</option>
                    <option value="sam_DZ">Dom zdravlja - Samobora</option>
                    <option value="ds_DZ">Dom zdravlja - Dugo Selo</option>
                </select><br>
            <label for="vrsta_dogadjaj">Vrsta događaja: </label>
                <select name="vrsta_dogadjaj">
                    <option value="Lom kosti na nespecifičnom mjestu">Lom 
kosti na nespecifičnom mjestu</option>
                    <option value="Modrica na nespecifičnom mjestu">Modrica 
na nespecifičnom mjestu</option>
                </select><br><br>
            <button type="submit">Pošalji</button>
    </div>
    <%- include("./partials/footer.ejs") %>
</body>
</html>
```

/views/analiticki_prikaz.ejs

```
<html lang="hr">
  <%- include("./partials/head.ejs") %>
  <body>
    <%- include("./partials/nav.ejs") %>
    <iframe
      src="http://neodash.graphapp.io/?share&type=database&id=d104d885-
8b89-4af8-b377-
f2559c869b03&dashboardDatabase=neo4j&credentials=neo4j%2Bs%3A%2F%2Fneo4j%3A
Eg7O6ak1ErjFPeCwsi-
IMcQds9gWKcQDDUx8jBN9mco%40%3A36e5e105.databases.neo4j.io%3A7687&standalone
=Yes"
      width="100%"
      height="1000"
      frameborder="0"
    ></iframe>

    <%- include("./partials/footer.ejs") %>
  </body>
</html>
```

NeoDash / Shema baze podataka

```
CALL db.schema.visualization()
```

NeoDash / Top 10 djece kod kojih postoji mogućnost zlostavljanja

```
MATCH (o1:Osoba)
WHERE (date().year - toInteger(o1.godiste)) < 18 AND o1.nadzor = "false"
OPTIONAL MATCH (o1)-[r1]->(ps:`ProblemŠkola`)
OPTIONAL MATCH (o1)-[r2]->(zdp:`ZdravstveniProblem`)
OPTIONAL MATCH (o1)-[r3]->(o2:Osoba)-[r4]->(zp:ZakonskiProblem)
OPTIONAL MATCH (o1)-[r5]->(o2:Osoba)-[r6]->(sp:SocijalniProblem)
RETURN o1.ime AS Ime, o1.prezime AS Prezime, o1.OIB AS OIB, count(distinct
r1)+count(distinct r2)+count(distinct r4)+count(distinct r6) AS Bodovi
ORDER BY Bodovi DESC
LIMIT 10
```

NeoDash / Grafički prikaz

```
MATCH (o1:Osoba) WHERE o1.OIB = $neodash_osoba_oib
OPTIONAL MATCH (o1)-[r1]->(o2:Osoba)
OPTIONAL MATCH (o1)-[r2]->(zdrp1:ZdravstveniProblem)-[r3]-
>(zdrp2:ZdravstvenaUstanova)
OPTIONAL MATCH (o1)-[r4]->(skolp1:ProblemŠkola)-[r5]-
>(skolp2:ŠkolskaUstanova)
OPTIONAL MATCH (o2)-[r10]->(zdrp3:ZdravstveniProblem)-[r11]-
>(zdrp4:ZdravstvenaUstanova)
OPTIONAL MATCH (o2)-[r12]->(skolp3:ProblemŠkola)-[r13]-
>(skolp4:ŠkolskaUstanova)
OPTIONAL MATCH (o2)-[r6]->(zakp1:ZakonskiProblem)-[r7]-
>(zakp2:PolicijskaPostaja)
OPTIONAL MATCH (o2)-[r8]->(socp1:SocijalniProblem)-[r9]-
>(socp2:SocijalnaSkrb)
OPTIONAL MATCH (o1)-[r14]->(zakp3:ZakonskiProblem)-[r15]-
>(zakp4:PolicijskaPostaja)
OPTIONAL MATCH (o1)-[r16]->(socp3:SocijalniProblem)-[r17]-
>(socp4:SocijalnaSkrb)
RETURN o1, o2, r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, r11, r12, r13, r14,
r15, r16, r17, zdrp1, zdrp2, skolp1, skolp2, zakp1, zakp2, socp1, socp2,
zdrp3, zdrp4, skolp3, skolp4, zakp3, zakp4, socp3, socp4
```