Prikupljanje i prikaz podataka s MikroTik mrežnih uređaja

Emšo, Marija

Master's thesis / Specijalistički diplomski stručni

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **The Polytechnic of Rijeka / Veleučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:125:257653

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-07-08**

Repository / Repozitorij:

Polytechnic of Rijeka Digital Repository - DR PolyRi

VELEUČILIŠTE U RIJECI

Marija Emšo

PRIKUPLJANJE I PRIKAZ PODATAKA S MIKROTIK MREŽNIH UREĐAJA

(specijalistički završni rad)

Rijeka, 2020.

VELEUČILIŠTE U RIJECI
Poslovni odjel
Specijalistički diplomski stručni studij
Informacijske tehnologije u poslovnim sustavima

PRIKUPLJANJE I PRIKAZ PODATAKA S MIKROTIK MREŽNIH UREĐAJA

(specijalistički završni rad)

MENTOR

Vlatka Davidović, viši predavač

STUDENT

Marija Emšo

MBS: 24220000961/18

Rijeka, rujan 2020.

VELEUČILIŠTE U RIJECI
Poslovni odjel

Rijeka, 4. lipnja 2020.

ZADATAK
za specijalistički završni rad

Pristupnica MARIJA EMŠO **MBS: 2422000096/18**

Studentici specijalističkog diplomskog stručnog studija Informacijske tehnologije u poslovnim sustavima izdaje se zadatak specijalističkog završnog rada – tema specijalističkog završnog rada pod nazivom:

PRIKUPLJANJE I PRIKAZ PODATAKA S MIKROTIK MREŽNIH UREĐAJA

Sadržaj zadatka: Razviti *web*-aplikaciju za prikupljanje podataka u realnom vremenu s MikroTik mrežnih uređaja. U aplikaciji osigurati obradu i vizualizaciju prikupljenih podataka. Opisati odabrane tehnologije, arhitekturu implementiranog sustava i programsko rješenje. Gotovu aplikaciju priložiti uz pisani dio rada.

Preporuka _____

Rad obraditi sukladno odredbama Pravilnika o završnom radu Veleučilišta u Rijeci.

Zadano: 4.lipnja 2020. **Predati do:** 15.rujna 2020.

Mentor:
Vlatka Davidović, viši predavač

Pročelnica odjela:
mr. sc. Anita Stilin, viši predavač

Zadatak primila dana: 4. lipnja 2020.

Marija Emšo

Dostavlja se:
- **mentoru**
- **pristupniku**

I Z J A V A

Izjavljujem da sam specijalistički završni rad pod naslovom PRIKUPLJANJE I PRIKAZ PODATAKA S MIKROTIK MREŽNIH UREĐAJA izradila samostalno pod nadzorom i uz stručnu pomoć mentora Vlatke Davidović, v. pred..

Ime i prezime

(potpis studenta)

SAŽETAK

U ovome će se specijalističkom završnom radu provesti izgradnja sustava za prikupljanje i prikaz mrežnih podataka, odnosno sustava za nadzor mreže. Istražiti će se tehnologije koje omogućuju izgradnju navedenog sustava te će se predstaviti funkcionalnosti koje jedan takav sustav mora sadržavati. Odgovoriti će se na pitanja: što je to MikroTik i čemu služi MikroTik API. Izraditi će se specifikacija zahtjeva na temelju kojih će se predstaviti popis funkcionalnosti koje sustav mora zadovoljiti. Grafički će se prikazati pojedini slučajevi korištenja sustava te pojedine aktivnosti kroz koje sustav prolazi. Opisati će se postupak razvoja kroz odabrane tehnologije. Odgovoriti će se na pitanja što je to NodeJS, Routeros-client, PHP i CodeIgniter. Dati će se pregled razvijenog rješenja i predstaviti daljnji koraci razvoja i implementacije sustava.

KLJUČNE RIJEČI: mreža, MikroTik, NodeJS, CodeIgniter, Routeros-client

SADRŽAJ

UVOD 1
1. OPIS SUSTAVA 2
2. MIKROTIK 3
3. SPECIFIKACIJA ZAHTJEVA 5
4. POPIS FUNKCIONALNOSTI 7
4.1. Dijagram slučaja korištenja 7
4.2. Dijagram aktivnosti 9
5. RAZVOJ RJEŠENJA 18
5.1.Arhitektura 18
5.2.Relacijski model podataka 22
6. KORIŠTENE TEHNOLOGIJE 24
6.1.PHP 24
6.1.1.CodeIgniter 25
6.1.2.AdminLTE 3.0.5 25
6.2.MySQL 29
6.3.NodeJS 30
6.3.1.Express 30
6.3.2.MySQL 31
6.3.3.Routeros-client 32
6.3.4.Socket.io 33
7. PREGLED KLIJENTSKOG DIJELA RJEŠENJA 35
8. DALJNJI RAZVOJ 46
9. ZAKLJUČAK 47
POPIS KORIŠTENIH KRATICA 48
POPIS LITERATURE 49
POPIS TABLICA 51
POPIS FOTOGRAFIJA 52
POPIS PROGRAMSKOG KODA 53

UVOD

Prikupljanje i prikaz mrežnih podataka je od iznimne važnosti za svaku djelatnost koja je uključena u mrežne i komunikacijske tehnologije. U slučajevima gdje mreža uključuje veći broj uređaja ta potreba postaje izraženija. Pregled i grafički prikaz podataka je neophodan za pravovremenu detekciju kvarova i upravljanje samom mrežom. Ponekad telekomunikacijske organizacije uz dolaznu vezu, krajnjim korisnicima omogućuju i ugradnju mrežnih uređaja koji su zaduženi za lokalnu mrežu. Razvojem mrežnih panela za pregled mrežnih podataka i uređaja u lokalnoj mreži korisnicima je omogućen nadzor vlastite mreže bez potrebe pristupa samim uređajima. Ovaj je rad posvećen upravo razvoju sustava i aplikativnog rješenja koje omogućuje nadzor i pregled mrežnih podataka lokalnih mrežnih uređaja. Autorica ovoga rada zaposlena je u mrežnom odjelu organizacije koja je uključena u projekt izgradnje pasivne infrastrukture i postavljanja aktivne mrežne opreme te idejne izrade aplikativnog rješenja za korisnički nadzor lokalne mreže u obrazovnim ustanovama. Obzirom da je autorica u dosadašnjem radu, najviše iskustva stekla na MikroTik mrežnim uređajima te se sama mreža u čiji je rad uključena sastoji od velikog broja uređaja navedenog proizvođača, izrada vlastitog rješenja koje omogućuje pregled mrežnih podataka i nadzor uređaja predstavljao je dodatan, praktičan izazov, za čiju je izradu potrebno uključiti programska i mrežna rješenja.

1. OPIS SUSTAVA

Panel za nadzor mreže namijenjen je administratorima mreže sastavljene od MikroTik mrežnih uređaja. Jedna od glavnih funkcionalnosti koje sustav mora zadovoljiti jest prikupljanje mrežnih podataka sa mrežnih uređaja. Neki od temeljnih podataka koji se prikupljaju u sustavima za nadzor mreže su podatci o samome uređaju, kao što je primjerice naziv, model i serijski broj uređaja, verzija operativnog sustava usmjerivača (engl. *Router Operating System* - ROS) te iskorištenost hardverskih resursa, zatim podatci o mrežnim parametrima i konfiguraciji uređaja, kao što su primjerice fizička i logička sučelja i postavljeni adresni rasponi (engl. *Internet Protocol address* – IP). Osim podataka o konfiguraciji, sustavi za nadzor mreže omogućuju i nadzor uređaja, primjerice kroz aktivno praćenje statusa pojedinih sučelja, kroz grafički prikaz dolazne i odlazne količine prometa te uz pomoć prikupljanja i prikaza različitih tipova zapisa (engl. *log*) koje je zabilježio uređaj.

Druga temeljna funkcionalnost koju sustav za upravljanje i nadzor mreže treba zadovoljiti jest mogućnost upravljanja mrežnim uređajima i korisnicima sustava, odnosno dodavanja novih mrežnih uređaja u nadzor u slučaju proširenja mreže te izmjene korisnika sustava u slučaju preraspodjele ljudskih resursa.

Dodatne funkcionalnosti koje bi sustav za nadzor i upravljanje mrežom mogao imati su pregled podataka o samome sustavu, kao što su primjerice podatci o aktivnoj bazi podataka i serveru te mogućnost tabličnog izvoza podataka. Sustav za upravljanje i nazor mrežnih uređaja ne treba omogućiti izmjenu konfiguracije na mrežnim uređajima, rekonfiguraciju uređaja je poželjno izvoditi putem terminala ili konzole, a ne putem *web*-aplikacije.

2. MIKROTIK

MikroTik je kompanija koja se bavi proizvodnjom usmjerivača i *wireless* sustava sa sjedištem u Latviji. MikroTik se na tržištu pojavljuje kao proizvod pristupačne cijene, manje hardverske kvalitete u odnosu na konkurentne proizvođače, kao što su primjerice CISCO ili Juniper, ali sa jednom posebnom odlikom koja ga čini primamljivim proizvodom davateljima usluga Internet-a (engl. *Internet Service Provider* - ISP), a to je jedinstveni programski sustav, RouterOS. MikroTik se kao kompanija, u početku bavila isključivo razvojem navedenog programskog sustava, no kasnije je prešla i u proizvodnju vlastitog hardvera pod nazivom RouterBOARD. (SIA Mikrotīkls, https://mikrotik.com/aboutus, (20.08.2020.))

Slika 1. MikroTik

Izvor 1: https://news-cdn.softpedia.com/images/fitted/620x348/MikroTik-Outs-RouterOS-6-20-Release-Candidate-for-All-Its-Devices.jpg, (28.08.2020.)

RouterOS je operativni sustav MikroTik hardvera, odnosno RouterBOARD-a, koji se temelji na Linux *kernel*-u verzije 3.3.5. Sam sustav moguće je postaviti i na računalo ili virtualnu mašinu ovisno o potrebi, te je isti dostupan na službenom *web*-sjedištu MikroTik kompanije. Ono što čini sustav jedinstvenim jest njegova fleksibilnost i mogućnost prilagodbe putem široke palete naredbi. Osim upravljanja temeljnim mrežnim postavkama on uključuje i dodatne funkcionalnosti, kao što su primjerice integrirane mogućnosti skriptiranja putem sučelja komandne linije (engl. *Command Line Interface* - CLI) te API servis (engl. *Application Programmable Interface* - API). (SIA Mikrotīkls, https://wiki.mikrotik.com/wiki/Manual:RouterOS_features, (20.08.2020))

API omogućuje korisnicima razvoj vlastitih programskih rješenja koja ostvaruju komunikaciju sa RoterOS-om, u svrhu prikupljanja informacija, izmjene konfiguracije te općenito upravljanja samim mrežnim uređajima. API slijedi sintaksu sučelja komandne linije te prema zadanim postavkama sa usmjerivačem ostvaruje komunikaciju po port-u 8728. Komunikacija se vrši slanjem enkodirane rečenice na što se dobiva jedna ili više rečenica kao odgovor. Svaka rečenica sastoji se od riječi, riječi se po njihovom sadržaju mogu podijeliti u pet tipova; komanda, atribut, API, upit i odgovor. Svaka rečenica započinje komandom, nakon koje slijedi atribut, a svaka je riječ enkodirana u obliku njene duljine i količine *byte*-a koju zauzima. Oznaka završetka rečenice je riječ terminacije ili nulta dužina riječi, odnosno 0x00 *byte*-ova. (SIA Mikrotīkls, https://wiki.mikrotik.com/wiki/Manual:API, (20.08.2020))

3. SPECIFIKACIJA ZAHTJEVA

Kao što je u 2. poglavlju navedeno, sustav za upravljanje i nadzor mreže mora zadovoljiti dvije temeljne funkcionalnosti; prikupljanje i prikaz podataka, te upravljanje korisnicima i uređajima u nadzoru. Poželjno bi bilo da sustav omogućuje i izvoz podataka te pregled podataka o samome sustavu. Na temelju navedenih zahtjeva izrađen je popis funkcionalnosti koje bi sustav trebao sadržavati, kategoriziran pomoću MoSCoW (engl. *Must have, Should havem Could have, Won't have* - MoSCoW) metode prioritizacije zahtjeva (tablica 2.) kojom se vrši kategorizacija pojedinih zahtjeva prema važnosti njihova ispunjenja:

1. *Must have* – Mora imati:
   - Prikupljanje mrežnih podataka sa uređaja
   - Prikaz prikupljenih mrežnih podataka
   - Unos/izmjena/brisanje mrežnih uređaja iz nadzora
   - Autentifikacija

2. *Should have* – Trebala bi imati
   - Grafički prikaz prikupljenih podataka
   - Upravljanje korisnicima

3. *Could have* – Može imati
   - Uvoz/izvoz podataka
   - Upravljanje postavkama
   - Spremanje zapisa

4. *Won't have* – Neće imati
   - Rekonfiguracija mrežnih uređaja
   - Direktno upravljanje mrežnim uređajima

Tablica 1. MoSCoW metoda analize zahtjeva

MUST HAVE	***SHOULD HAVE***
• Prikupljanje mrežnih podataka sa uređaja • Prikaz prikupljenih mrežnih podataka • Unos/izmjena/brisanje mrežnih uređaja iz nadzora • Autentifikacija	• Grafički prikaz prikupljenih podataka • Upravljanje korisnicima
COULD HAVE	***WON'T HAVE***
• Uvoz/izvoz podataka • Upravljanje postavkama • Spremanje zapisa	• Rekonfiguracija mrežnih uređaja • Direktno upravljanje mrežnim uređajima

Izvor 2: Autorica

4. POPIS FUNKCIONALNOSTI

Na temelju zahtjeva koji su specificirani u poglavlju 4. te uz pomoć dijagrama slučaja korištenja izrađen je pregled pojedinih slučajeva korištenja kroz koje korisnici budućeg sustava prolaze. Također su, uz pomoć dijagrama aktivnosti, predstavljene pojedine aktivnosti koje sustav izvršava ovisno o predstavljenim zahtjevima i slučajevima korištenja. Prikaz i opis pojedinih dijagrama slijedi u nastavku poglavlja.

4.1.Dijagram slučaja korištenja

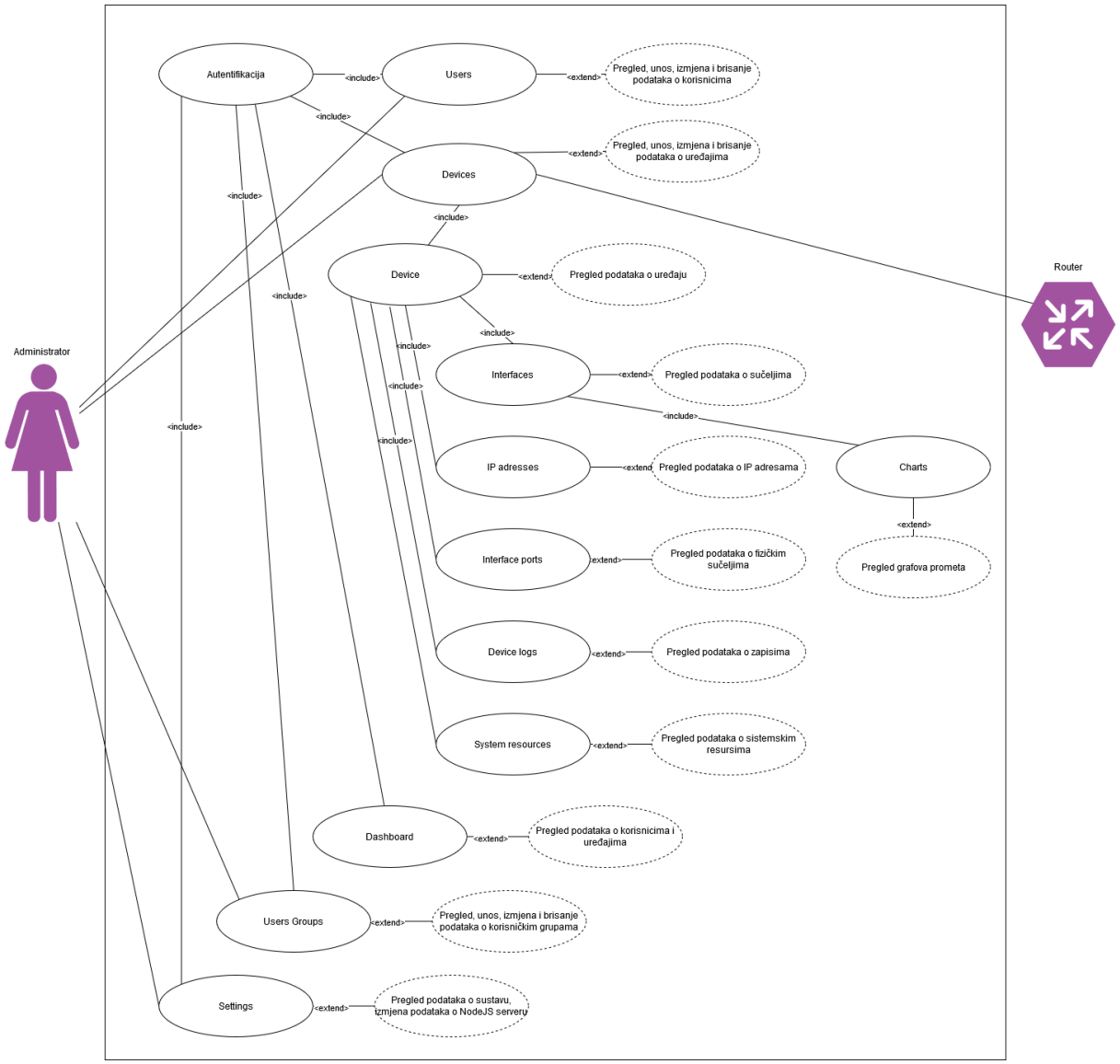
Slika 2. prikazuje aktivnosti aktera sustava u obliku UML dijagrama korištenja (engl. *Unified Modeling Language* - UML). Vidljivo je da sustav koriste dva aktera, „*Administrator*“ i „*Router*“. „*Administrator*“ sudjeluje u svim slučajevima korištenja koje sustav sadržava, dok „*Router*“ pristupa samo uređajima, odnosno „*Devices*“ aktivnosti.

Slučajevi korištenja: „*Users*“, „*Devices*“, „*Dashboard*“, „*User Groups*“ i „*Settings*“ su uključeni, „<*include*>“ u slučaju „*Autentifikacija*“, dok slučaj „*Devices*“ dodatno uključuje slučaj „*Device*“ koji uključuje slučaje korištenja: „*Interfaces*“, „*IP addresses*“, „*Interface ports*“, „*Device logs*“ i „*System resources*“. Slučaj „*Interfaces*“ uključuje slučaj korištenja „*Charts*“.

Slučaj „*Users*“ se proširuje, „<*extend*>“ u „*Pregled, unos, izmjena i brisanje podataka o korisnicima*“, slučaj „*Devices*“ se proširuje u „*Pregled, unos, izmjena i brisanje podataka o uređajima*“, slučaj, „*Device*“ se proširuje u „*Pregled podataka o uređaju*“, slučaj „*Interfaces*“ se proširuje u „*Pregled podataka o sučeljima*“, dok se slučaj „*Charts*“ proširuje u „*Pregled grafova prometa*“. Slučaj „*IP adresses*“ se proširuje u „*Pregled podataka o sučeljima*“, slučaj „*Interface ports*“ se proširuje u „*Pregled podataka o fizičkim sučeljima*“, slučaj „*Device logs*“ se proširuje u „*Pregled podataka o zapisima*“ dok se slučaj „*System resources*“ proširuje u „*Pregled podataka o sistemskim resursima*“. Slučaj korištenja „*Dashboard*“ se proširuje u

„*Pregled podataka o korisnicima i uređajima*“, slučaj „*User Goups*“ se proširuje u „*Pregled, unos, izmjena i brisanje podataka o korisničkim grupama*“ dok se slučaj „*Settings*“ se proširuje u „*Pregled podataka o sustavu i izmjena podataka o NodeJs serveru*“.

Slika 2. Dijagram slučaja korištenja

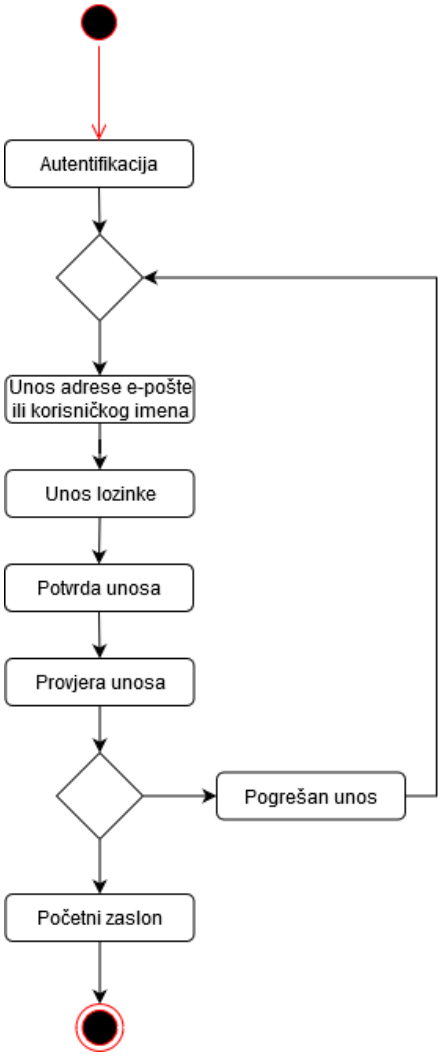


Izvor 3: Autorica

4.2.Dijagram aktivnosti

Za opis pojedinih scenarija korišten je dijagram UML dijagram aktivnosti pomoću kojeg se prikazuje tok izvođenja aktivnosti te uvjeti i kontrole izvođenja pojedinih aktivnosti.

Slika 3. Dijagram aktivnosti – „Autentifikacija“



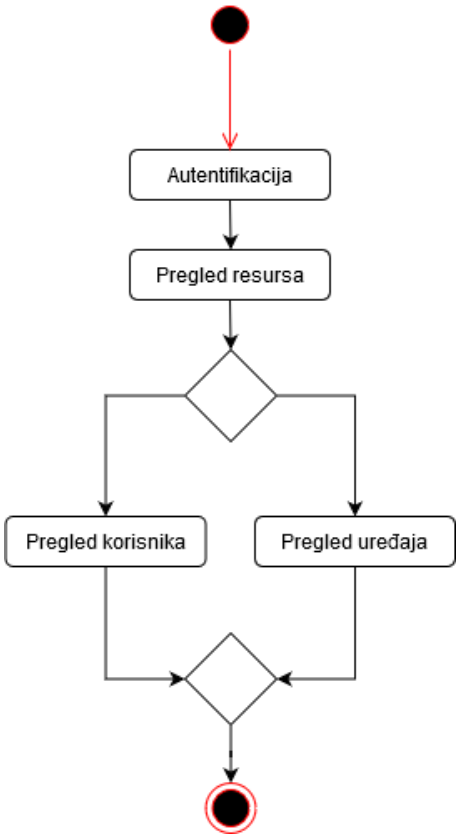
Izvor 4: Autorica

Na slici 3. je prikazan dijagram aktivnosti za slučaj „*Autentifikacija*“. Prilikom pristupa *web*-aplikaciji, administratoru, odnosno korisniku aplikacije prvo se javlja autentifikacijski

zaslon. Korisnik unosi pristupne podatke; adresu e-pošte te lozinku, ukoliko su podatci ispravni otvara se početni, upravljački zaslon web-aplikacije, no ukoliko su podatci pogrešno uneseni, otvara se početni zaslon za prijavu u sustav sa upozorenjem o pogrešnom unosu.

Na slici 4. je prikazan dijagram aktivnosti za slučaj „*Dashboard*“. Prije samog pristupa upravljačkom panelu potrebno se autentificirati u sustav. Nakon uspješne autentifikacije otvara se upravljački panel sa podatcima o ukupnom broju aktivnih i neaktivnih uređaja u nadzoru te podatcima o ukupnom broju korisnika sustava.

Slika 4. Dijagram aktivnosti – „*Dashboard*“



Izvor 5: Autorica

Na slici 5. je prikazan dijagram aktivnosti za slučaj „*Users*“. Prije samog pristupa korisnicima, potrebna je autentifikacija u sustav. Nakon uspješne autentifikacije administrator može pristupiti korisnicima gdje su mu vidljivi osnovni podatci o svim korisnicima sustava.

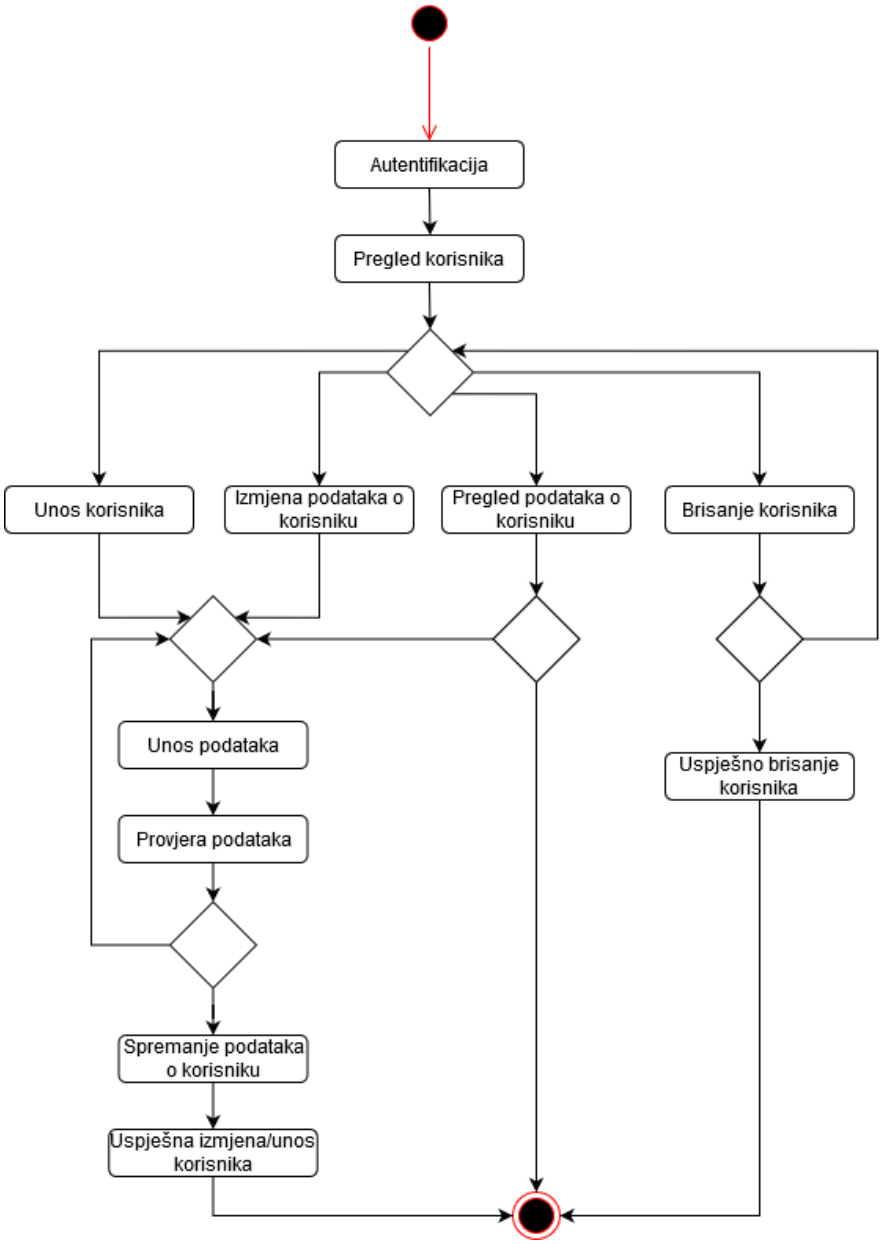
Administrator tada može odabrati unos novog korisnika u sustav, tada se otvara zaslon sa formom za unos podataka o korisniku. Ukoliko su podatci pogrešno uneseni, odnosno ukoliko adresa e-pošte nije ispravno unesena ili su pak uneseni podatci o korisničkom imenu i adresi e-pošte koji već postoje u bazi, unos nije uspješan te se na zaslonu sa unos korisnika javlja upozorenje o pogrešnom ili duplom unosu. Ukoliko su podatci ispravno uneseni, korisnik se dodaje u bazu podataka te se otvara zaslon sa prikazom osnovnih podataka o svim korisnicima zajedno sa podatcima novo-unesenom korisniku te obavijesti da su podatci uspješno izmijenjeni.

Ukoliko administrator odabere izmjenu korisnika, otvara se zaslon sa formom za izmjenu podataka o korisniku. Ukoliko su podatci pogrešno uneseni, odnosno ukoliko adresa e-pošte nije ispravno unesena ili su uneseni podatci o e-pošti ili korisničkom imenu koji već postoji u bazi, izmjena nije primijenjena te se javlja upozorenje o pogrešnom ili duplom unosu. Ukoliko su podatci ispravno uneseni, primjenjuje se izmjena korisničkih podataka u bazu podataka te se otvara zaslon sa prikazom osnovnih podataka o svim korisnicima zajedno sa izmijenjenim podatcima o korisniku te obavijesti da su podatci uspješno izmijenjeni.

Odabirom pregleda korisnika, otvara se zaslon sa detaljima o profilu korisnika. Administrator prilikom pregleda detalja o profilu korisnika može odabrati izmjenu podataka o odabranome korisniku.

Za slučaj brisanja korisnika, otvara se prozor sa upozorenjem i odabirom želi li se zaista izbrisati korisnik iz baze. Tada administrator može odabrati prekid, čime se vraća na zaslon sa pregledom osnovnih podataka o korisnicima ili brisanje, čime se brišu podatci o korisniku iz baze podataka te se javlja obavijest da je korisnik uspješno izbrisan.

Slika 5. Dijagram aktivnosti – „*Users*“



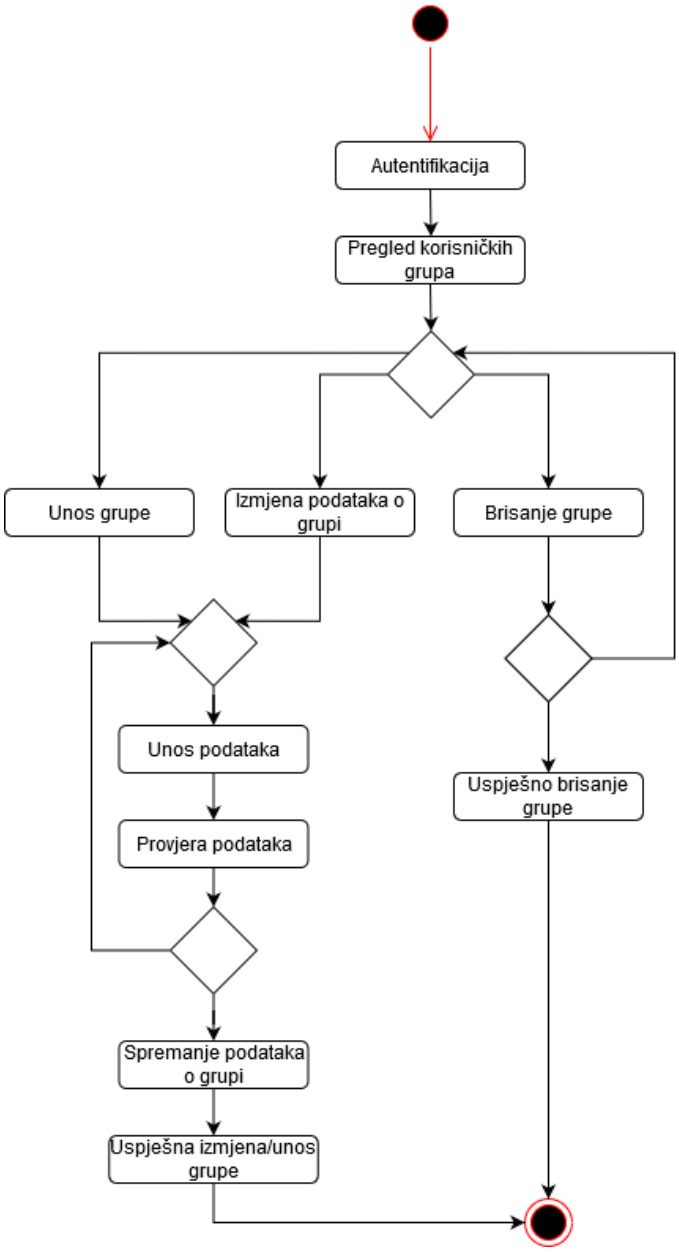
Izvor 6: Autorica

Na slici 6. je prikazan dijagram aktivnosti za slučaj „*Users Groups*“. Prije samog pristupa korisničkim grupama, potrebna je autentifikacija u sustav. Nakon uspješne autentifikacije administrator može pristupiti korisničkim grupama gdje su mu vidljivi podatci o postojećim korisničkim grupama.

Prilikom odabira unosa nove korisničke grupe u sustav, otvara se zaslon sa formom za unos podataka o korisničkim grupama. Ukoliko su podaci ispravno uneseni, korisnička grupa se sprema u bazu podataka te se prikazuje zaslon sa pregledom korisničkih grupa zajedno sa novo-unesenom korisničkom grupom i obavijesti da je korisnička grupa uspješno unesena. Ukoliko su podaci krivo uneseni, odnosno unesen je naziv korisničke grupe koji već postoji u bazi podataka, grupa se ne unosi u bazu podataka te se javlja upozorenje da se uneseni naziv grupe već koristi.

U slučaju brisanja korisničke grupe javlja se prozor sa upitom želi li se nastaviti sa brisanjem. U slučaju da administrator odustaje od brisanja vraća se na zaslon sa prikazom podataka o korisničkim grupama, u suprotnom, dakle ako odabere brisanje, korisnička grupa se briše iz baze podataka te se javlja obavijest da je korisnička grupa uspješno izbrisana.

Slika 6. Dijagram aktivnosti – „*Users Groups*“



Izvor 7: Autorica

Na slici 7. je prikazan dijagram aktivnosti za slučaj „*Devices*“. Prije samog pristupa uređajima, potrebna je autentifikacija u sustav. Nakon uspješne autentifikacije administrator može pristupiti uređajima gdje su mu vidljivi osnovni podatci o postojećim uređajima.

Odabirom unosa novog uređaja otvara se zaslon sa formom za unos podataka o uređaju. Administrator unosi podatke o uređaju te se vrši provjera pogrešnih, duplih ili nepotpunih unosa, odnosno ukoliko je unesen krivi format IP adrese ili port-a, javlja se upozorenje da je podatak krivo unesen, ukoliko nisu uneseni podatci o IP adresi, korisničkom imenu i lozinki javlja se upozorenje o obaveznom unosu, te ukoliko su podatci o IP adresi i API port-u već uneseni u bazi, javlja se upozorenje o postojećem uređaju. Ukoliko su podatci ispravno uneseni, uređaj se dodaje u bazu podataka te se otvara zaslon sa podatcima o uređajima, zajedno sa obavijesti da je uređaj uspješno unesen. Prije samog unosa podataka o uređaju, administrator može izvršiti provjeru konekcije. Ukoliko se konekcija prema unesenim parametrima uspostavi, korisniku se javlja obavijest da je konekcija uspostavljena, u suprotnom javlja se upozorenje da je vrijeme za konekciju isteklo.

Prilikom izmjene podataka o uređaju, otvara se zaslon sa formom za izmjenu podataka o uređaju. Podatci se izmjenjuju te se vrši provjera izmijenjenih podataka kao što je to slučaj kod unosa podataka o uređaju. Prije izmjene podataka moguće je, kao i kod unosa podataka o uređaju, napraviti provjeru konekcije. Ukoliko su podatci ispravno uneseni, otvara se zaslon sa podatcima o uređajima zajedno sa obavijesti o uspješnoj izmjeni.

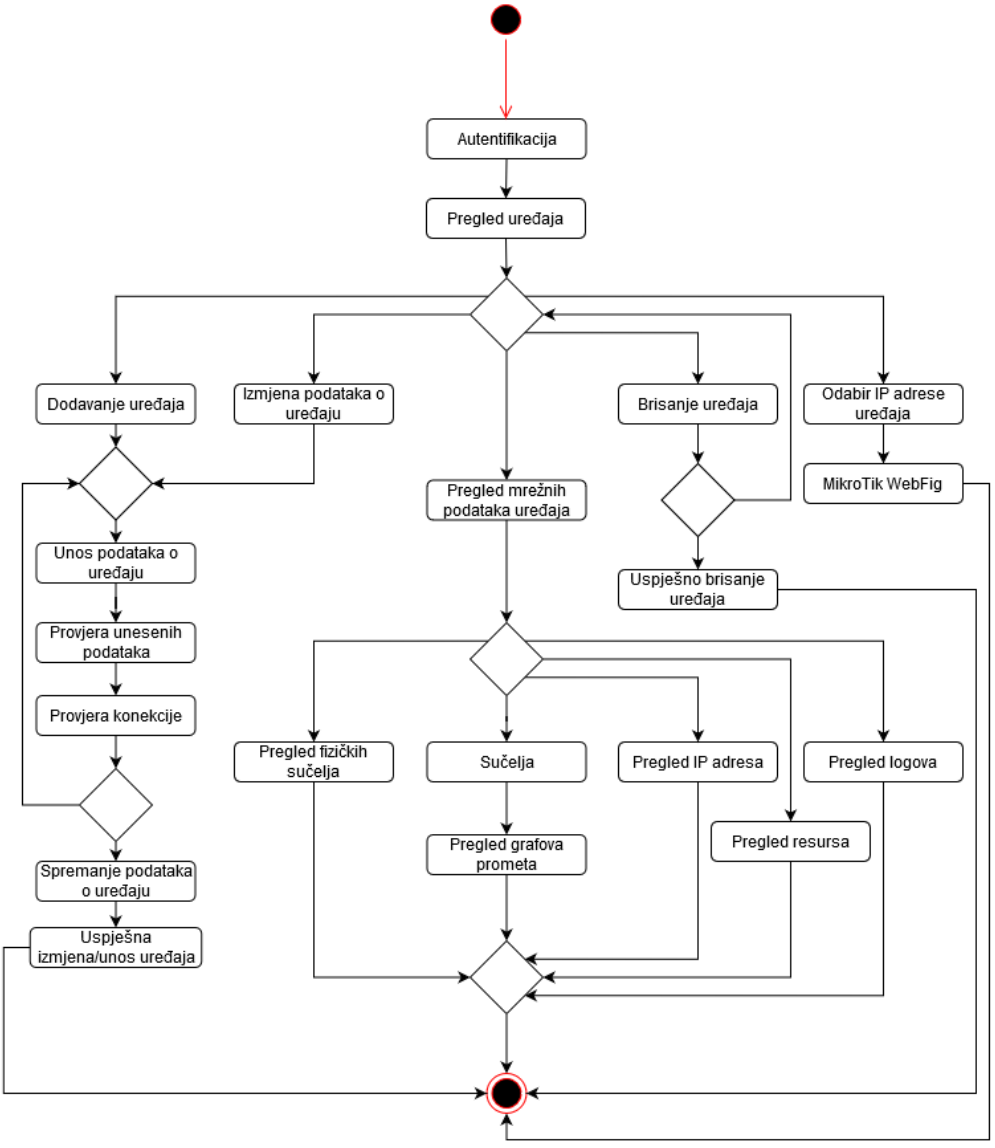
U slučaju brisanja uređaja, javlja se prozor sa upozorenjem, ukoliko se odabere prekid, otvara se zaslon sa prikazom podataka o uređajima, u suprotnom, dakle ako se odabere brisanje uređaja, uređaj se briše i baze podataka te se javlja obavijest da je uređaj uspješno izbrisan.

Ukoliko administrator na zaslonu prikaza podataka o uređaju klikne na poveznicu, odnosno IP adresu uređaja, otvara se MikroTik WebFig[1] odabranog mrežnog uređaja.

[1] *WebFig is a web based RouterOS utility which allows you to monitor, configure and troubleshoot the router. WebFig is accessible directly from the router which means that there is no need to install additional software (except web browser with JavaScript support). As Webfig is platform independent, it can be used to configure router directly without need of a software developed for specific platform.* (SIA Mikrotīkls, https://wiki.mikrotik.com/wiki/Manual:Webfig (28.08.2020.))

Ukoliko administrator odabere pregled podataka o mrežnom uređaju, otvara se zaslon sa detaljnim podatcima o odabranome mrežnom uređaju, odnosno pregled podataka o definiranim logičkim sučeljima, pregled podataka o definiranim IP adresama, pregled podataka o fizičkim sučeljima, pregled podataka o zapisima sa uređaja te pregled podataka o resursima uređaja. Ukoliko administrator prilikom pregleda podataka o logičkim sučeljima, odabere prikaz detalja o logičkom sučelju otvara se zaslon sa grafom prikaza trenutnog prometa odabranog sučelja.

Slika 7. Dijagram aktivnosti – „*Devices*“

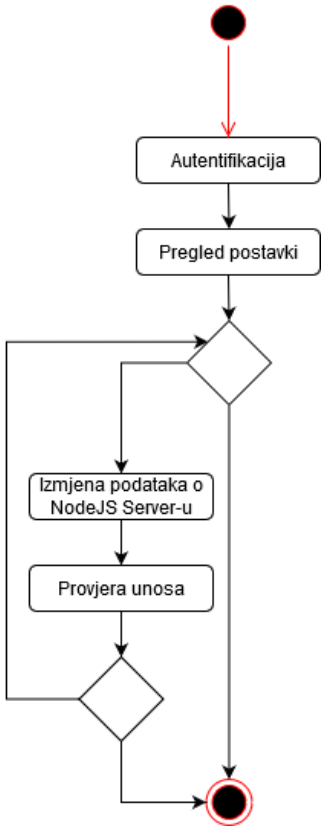


Izvor 8: Autorica

Na slici 8. je prikazan dijagram aktivnosti za slučaj „*Settings*“. Prije samog pristupa postavkama, potrebna je autentifikacija u sustav. Nakon uspješne autentifikacije administrator može pristupiti postavkama gdje su mu vidljivi osnovni podatci o postavkama sustava.

Ako administrator želi izmijeniti podatke o NodeJS poslužitelju, on unosi podatke u formu za izmjenu. Ako su uneseni podatci neispravni, odnosno ukoliko je *port* netočno unesen, javlja se upozorenje da je unos neispravan, u suprotnom podatci se mogu spremiti, čime se izmjena sprema u bazu podataka te se javlja obavijest da je izmjena uspješno izvršena.

Slika 8. Dijagram aktivnosti – „*Settings*“



Izvor 9: Autorica

5. RAZVOJ RJEŠENJA

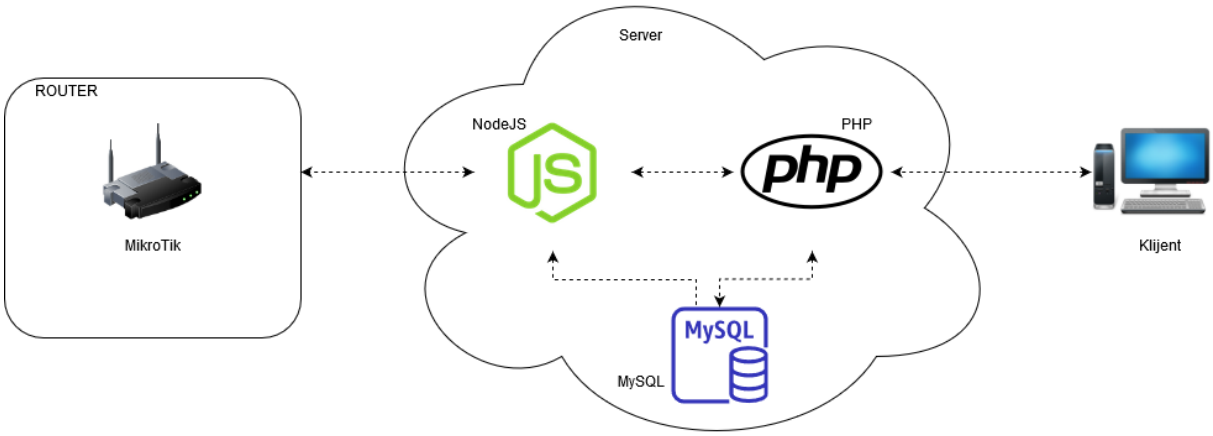
Na temelju specificiranih funkcionalnih zahtjeva, definiranih slučajeva korištenja i pojedinih aktivnosti stvoreno je sustavno, mrežno i programsko arhitektonsko rješenje izgradnje sustava te je izrađen relacijski model podataka buduće baze podataka. Pregled pojedinih rješenja za navedene slojeve sustava slijedi u nastavku poglavlja.

5.1.Arhitektura

Na slici 9. je prikazana arhitektura koja je korištena prilikom razvoja sustava za prikupljanje i prikaz mrežnih podataka sa uređaja. Sustav se sastoji tri temeljne komponente; API koji ostvaruje konekciju sa mrežnim uređajem te prikuplja sa njega podatke, *web*-aplikacije koja služi za pregled prikupljenih podataka i administraciju samih uređaja i korisnika sustava, te lokalne mreže koja se sastoji od mrežnih uređaja sa kojih se navedeni podatci prikupljaju. Za izradu API dijela odabran je NodeJS, dok je za izradu *web*-aplikacije odabran PHP programski jezik, odnosno CodeIgniter programski okvir. Za izradu baze podataka odabran je MySQL sustav za upravljanje bazama podataka dok je kao mrežni uređaj sa kojeg se prikupljaju mrežni podatci odabran MikroTik. NodeJS, PHP i MySQL se nalaze na istom poslužitelju.

Korisnik šalje zahtjev putem *web*-aplikacije, PHP zahtjev šalje na NodeJS koji dohvaća pristupne podatke za uređaj iz MySQL baze te se povezuje na MikroTik mrežni uređaj, zadaje mu upit te prima odgovor koji se vraća PHP-u i prikazuje korisniku. Korisnik također može unositi, izmijeniti ili brisati podatke, odnosno PHP sprema izmjene u MySQL bazu podataka te NodeJS potom koristi pristupne podatke za uspostavu komunikacije sa MikroTik mrežnim uređajem.

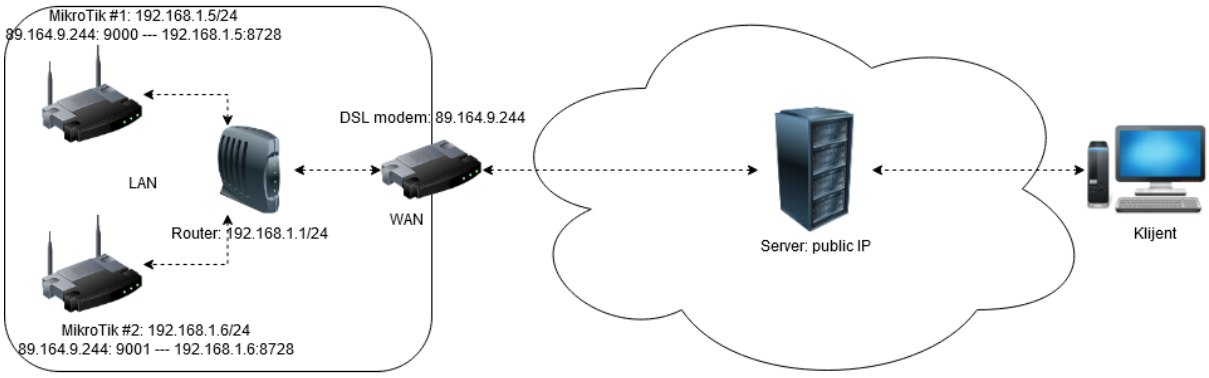
Slika 9. Shematski prikaz arhitekture sustava



Izvor 10: Autorica

Na slici 10. je prikazana pojednostavljena mrežna topologija koja je korištena za razvoj sustava za prikupljanje i prikaz mrežnih podataka. Obzirom da sustav treba omogućiti nadzor većeg broja uređaja, korištena su dva MikroTik mrežna uređaja; RouterBOARD cAP-2nD te 911G-5HPnD. Navedeni uređaji spajaju se na usmjerivač koji je povezan na DSL (engl. *Digital Subscriber Line* - DSL) modem. Na mrežnom usmjerivaču postavljen je IP adresni raspon 192.168.1.1/24. MikroTik uređaju cAP-2nD dodijeljena je adresa 192.168.1.5 te je napravljeno prosljeđivanje sa *port*-a 9000 na zadani MikroTik API *port* 8728. MikroTik uređaju 911G-5HPnD dodijeljena je adresa 192.168.1.6 te je napravljeno prosljeđivanje sa port-a 9001 na zadani MikroTik API port 8728. Dakle svi upiti koji pristižu sa javne adrese 89.164.9.244 po *port*-u 9000 prosljeđuju se na privatnu adresu 192.168.1.5 na *port* 8728, a svi upiti koji pristižu sa javne adrese 89.164.9.244 po port-u 9001 prosljeđuju se na privatnu adresu 192.168.1.6 na *port* 8728.

Slika 10. Shematski prikaz mrežne arhitekture

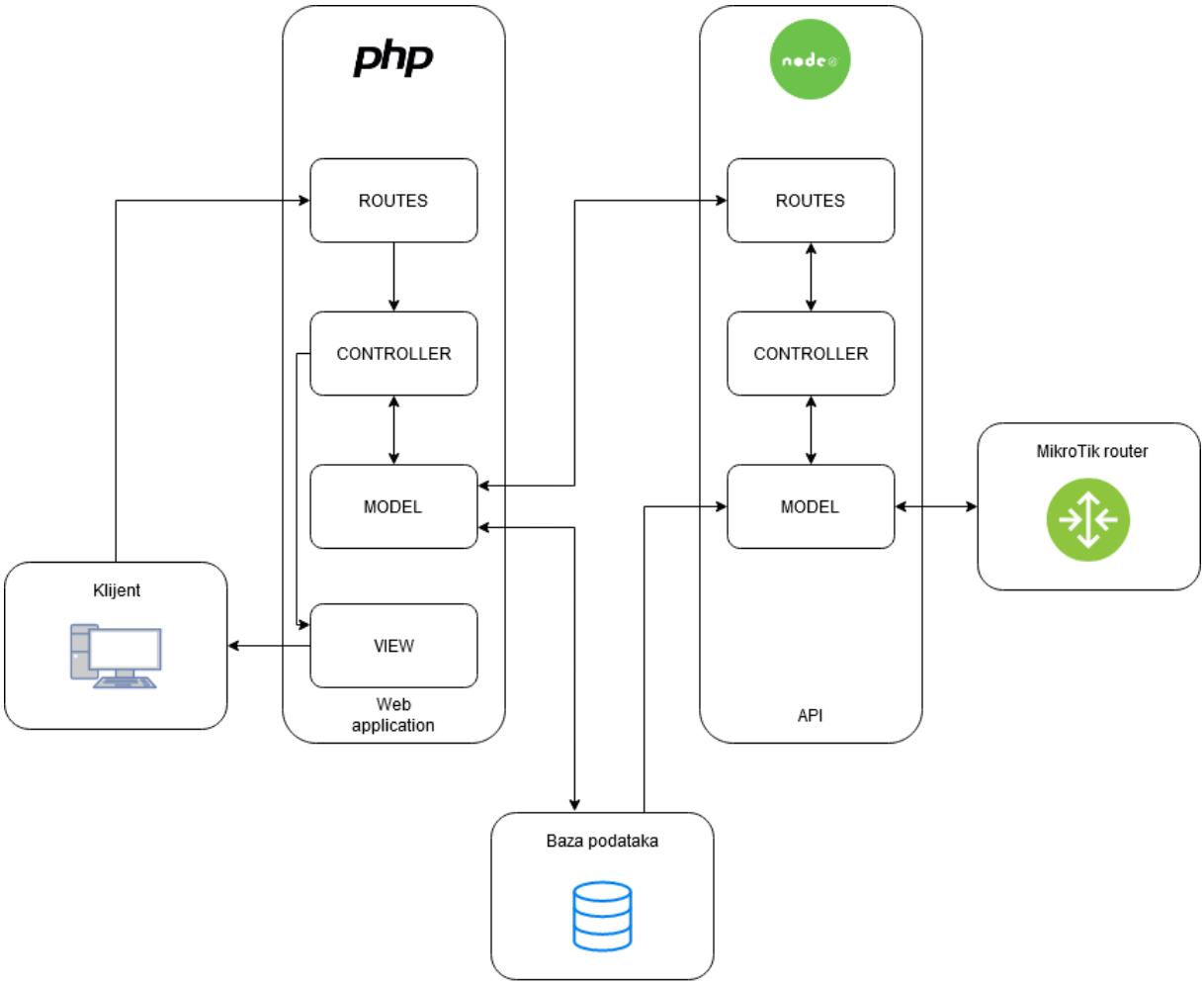


Izvor 11: Autorica

Na slici 11. je prikazana programska arhitektura koja je korištena pri sustava za prikupljanje i prikaz mrežnih podataka. Kao što je prethodno navedeno, sustav se sastoji od tri glavne komponente; *web*-aplikacije, API-a te mrežnih uređaja. Za izradu *web*-aplikacije korišten je PHP programski okvir CodeIgniter dok je za izradu API-a korišten NodeJS. Prilikom izrade projekta nastojala se pratiti MVC arhitektura (engl. *Model-view-controller* - MVC) razvoja, odnosno programski okvir CodeIgniter predefinirano koristi raspodjelu komponenti prema MVC arhitekturi, dok je pri izradi NodeJS API-a takvu podjelu autorica sama stvorila.

MVC arhitektura olakšava izradu, održavanje i pretraživanje koda. *Model* je zadužen za upravljanje podatcima, *view* za prikaz informacija krajnjem korisniku dok je *controller* poveznica koja omogućuje prijenos podataka između *model*-a i *view*-a. *View* dobiva podatke u obliku varijabli od *controller*-a, *controller* prihvaća unose korisnika te ih prosljeđuje u *model* koji ih sprema, *controller* također šalje zahtjeve za dohvat podataka *model*-u i potom ih prosljeđuje u *view*. *Model*, kao što je prethodno navedeno, čuva podatke, odnosno sprema i dohvaća podatke iz baze podataka.

Slika 11. Shematski prikaz programske arhitekture



Izvor 12: Autorica

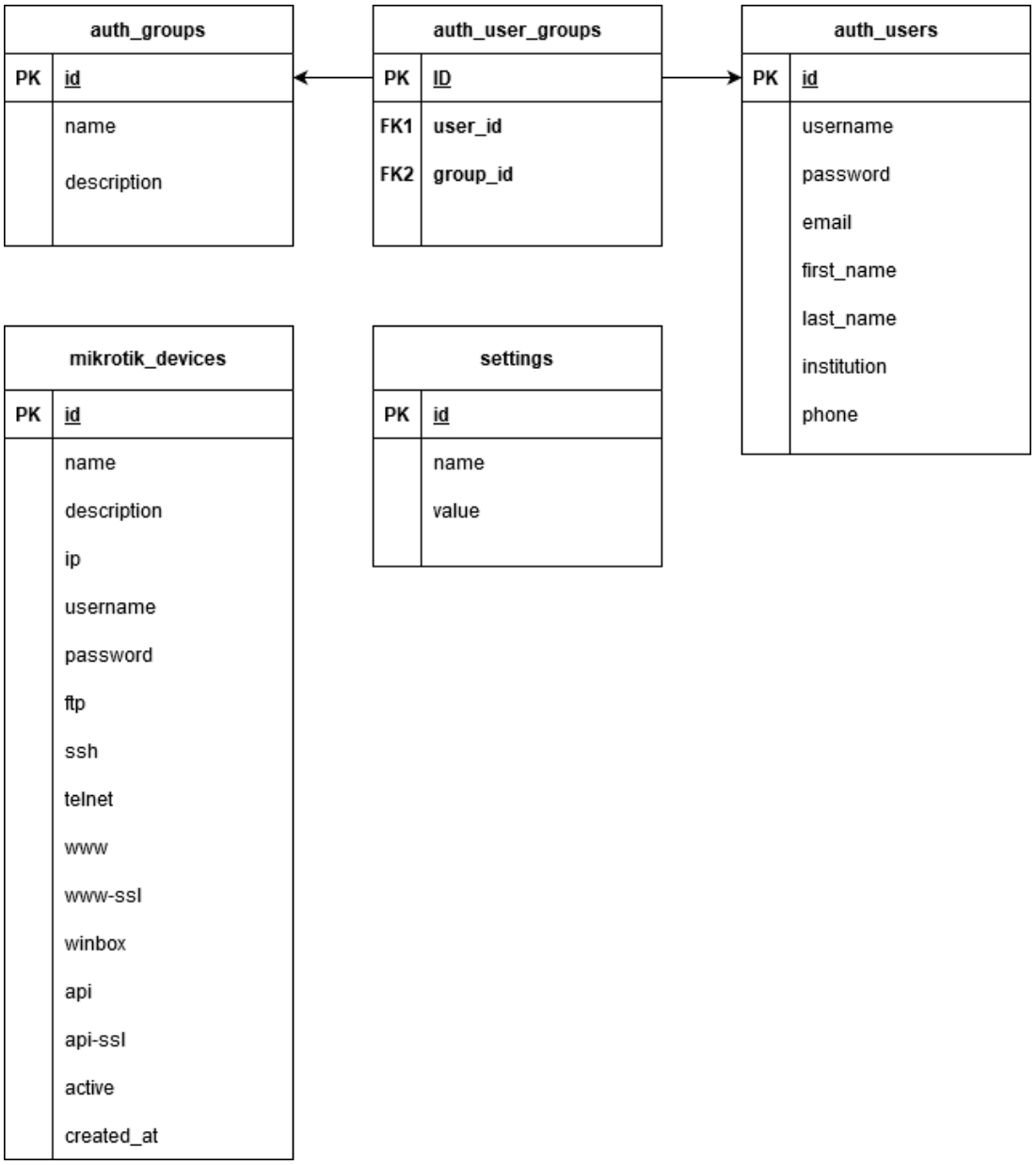
Korisnik sustava za prikupljanje i prihvat mrežnih podataka pošalje upit, upit pomoću PHP *routes* putuje u PHP *controller*, PHP *controller* šalje zahtjev u PHP *model* koji se, u slučaju da se radi o podatcima koji su pohranjeni u bazi podataka, spaja na bazu, dohvaća ih, i šalje natrag u PHP *controller* koji ih potom preko *view*-a prikazuje korisniku. Ukoliko je PHP *model* zaprimio zahtjev za podatcima koji se nalaze na mrežnom uređaju, tada on šalje upit preko NodeJS *routes*, na NodeJS *controller*, NodeJS *controller* šalje upit u NodeJS *model*, NodeJS *model* dohvaća pristupne podatke iz baze podataka, spaja se na MikroTik, vraća dohvaćene mrežne podatke u NodeJS *controller*, koji preko NodeJS *routes* vraća iste u PHP *model*. PHP *model* vraća mrežne podatke u PHP *controller* koji ih potom prosljeđuje u *view* do korisnika.

5.2.Relacijski model podataka

Na slici 12. je prikazan relacijski model podataka koji je korišten prilikom izrade baze podataka u MySQL sustavu za upravljanje bazom podataka. Baza sadrži pet tablica podataka: „*auth_groups*“, „*auth_user_groups*“, „*auth_users*“, „*mikrotik_devices*“ te „*settings*“. Tablica „*auth_groups*“ sadrži podatke o korisničkim grupama, tablica „*auth_users*“ sadrži podatke o korisnicima sustava, dok tablica „*auth_user_groups*“ čini agregaciju tih dviju tablica te omogućuje spremanje podataka o korisničkim grupama pojedinih korisnika. U navedene tablice podatke spremaju korisnici *web*-aplikacije prilikom unosa novih korisnika i korisničkih grupa te se ti podatci koriste samo u *web*-aplikaciji za prikaz podataka i autentifikaciju u sustav.

Tablica „*mikrotik_devices*“ sadrži podatke o uređajima u nadzoru. Podatke također unose korisnici, međutim, osim za prikaz podataka putem *web*-aplikacije, ti se podatci koriste za povezivanje NodeJS API-a na pojedini mrežni uređaj. Tablica „*settings*“ sadrži podatke o portu i adresi NodeJS poslužitelja, te podatke također unose i izmjenjuju korisnici putem *web*-aplikacije.

Slika 12. Relacijski model baze podataka



Izvor 13: Autorica

6. KORIŠTENE TEHNOLOGIJE

Na temelju specificiranih funkcionalnih zahtjeva, definiranih slučaja korištenja i aktivnosti odvijanja sustava te razrade sustavne, mrežne i programske arhitekture i strukture baze podataka odabrane su tehnologije razvoja te je izrađeno je programsko rješenje sustava za prikupljanje i prikaz mrežnih podataka. Kratak opis i prikaz korištenja pojedinih tehnologija slijedu u nastavku poglavlja.

6.1.PHP

Za izradu *web* aplikacije korišten je PHP programski jezik verzije 7.4 (engl. *Hypertext Preprocessor* - PHP), odnosno CodeIgniter programski okvir (engl. *framework*) verzije 3.11. PHP je skriptni jezik otvorenoga koda i opće namjene pogodan za razvoj *web*-a.

PHP je usredotočen na poslužiteljsku stranu, što znači da ima mogućnosti većine CGI programa (engl. *Common Gateway Interface* - CGI), kao što je prikupljanje podataka, generiranje dinamičkog sadržaja ili slanje i primanje kolačića. Dakle, skripta se izvršava na poslužiteljskoj strani, generira se, primjerice HTML (engl. *HyperText Markup Language* – HTML) dokument, koji se potom šalje na klijentsku stranu.

PHP je kompatibilan sa svim vodećim operativnim sustavima kao što su Linux, većina Unix izvedbi, Microsoft Windows, macOS, RISC OS, itd. (i tako dalje). Podržan je i od strane većine *web* poslužitelja, što uključuje Apache, IIS, NGINX, i mnoge druge. Također je podržan i od strane velikog broja baza podataka, kao što su primjerice MySQL, PostgreSQl, Ingres, MongoDB, itd. (Achour et al., https://www.php.net/manual/en/, (28.08.2020.))

6.1.1.CodeIgniter

Kao što je u prethodnom poglavlju (6.1.) navedeno, za izradu ovoga projekta korišten je CodeIgniter programski okvir verzije 4.0.4. CodeIgniter je okvir za izradu aplikacija, odnosno RAD alat (engl. *Rapid Application Development* - RAD) za razvoj *web*-aplikacija u PHP programskom jeziku temeljen na MVC arhitekturi razvoja. Cilj programskog okvira općenito jest omogućiti brži razvoj aplikacija pomoću raznih biblioteka prilagođenih za izvođenje predefiniranih, ponavljajućih zadataka. CodeIgniter koristi minimalne količine memorijskih resursa te je kompatibilan sa MySQL, PostgreSQL i SQLite3 bazama podataka i PHP verzijama od 7.2 na dalje. (CodeIgniter Foundation, https://codeigniter4.github.io/userguide/intro/index.html, (28.08.2020.))

Programski kod 1. Primjer korištenja CodeIgniter insert funkcije

```php
final public function insert_new_device(): ?int {

    $data = $this->input->post();

    $insert_arr = [
        'name'          => $data['name'],
        'description'   => $data['description'],
        'ip'            => $data['ip'],
        'username'      => $data['username'],
        'password'      => $data['password'],
        'active'        => 0,
        'created_at'    => date('Y-m-d H:i:s')
    ];

    $this->db->insert(self::TABLE_NAME, $insert_arr);
    $id = $this->db->insert_id();

    return ($id) ? $id : NULL;
}
```

Izvor 14: Autorica

6.1.2.AdminLTE 3.0.5

Za izradu dizajna *web*-aplikacije korišten je AdminLTE predložak (engl. *template*) verzije 3.0.5 koji se temelji na Bootstrap okviru verzije 4.4. AdminLTE je responzivan predložak prilagođen izradi administratorskih kontrolnih ploča (engl. *dashboard*) i panela.

Osnovna verzija predloška je besplatna i otvorenoga koda, dok se dodatni predlošci naplaćuju. (Colorlib, https://github.com/ColorlibHQ/AdminLTE, (28.08.2020.))

Bootstrap je CSS (engl. *Cascading Style Sheets* - CSS), HTML i JavaScript okvir otvorenoga koda za razvoj klijentskog dijela aplikacije (engl. *Frontend*). Kompatibilan je sa svim *web*-preglednicima osim sa Internet Explorer-om verzije 9. Bootstrap sadrži predloške temeljene na HTML-u i CSS-u za razvoj uobičajenih UI (engl. *User Interface* - UI) komponenata kao što su dugmad, padajući izbornici, tablice, navigacijske trake, itd., te JavaScript ekstenzije koje su ovisne o jQuery-u i Popper.js-u. (Otto et al., https://getbootstrap.com/docs/4.5/getting-started/introduction/, (28.08.2020.)) Dakle, osim o Bootstrap-u, AdminLTE je ovisan i o jQuery-u verzije 3.3.1 na dalje te o Popper.js-u verzije 1.14.7 na dalje. (Colorlib, https://adminlte.io/docs/3.0/dependencies.html, (28.08.2020.)) Popper.js je JavaScript biblioteka koja olakšava upravljanje, odnosno postavljanje *tooltip* ili *popover* UI elemenata. (Zivolo, https://popper.js.org/docs/v2/, (28.08.2020.))

JQuery je JavaScript biblioteka koja pojednostavljuje korištenje JavaScript-a. Korištenje JavaScript-a je pojednostavljeno na način da jQuery oblikuje (engl. *wrap*) najčešće korištene funkcije JavaScript-a, za koje je potrebno napisati veće količine koda, u metode koje se potom pozivaju pomoću jedne linije koda. JQuery uključuje HTML i CSS manipulaciju, animacije, HTML *event* metode te AJAX (engl. *Asynchronous JavaScript and Extensible Markup Language* - AJAX). Dakle, jQuery olakšava izvođenje složenijih radnji kao što su AJAX pozivi i DOM manipulacija (engl. *Document Object Modeling* - DOM). (The jQuery Foundation, https://jquery.com/, (28.08.2020.))

Programski kod 2. Primjer korištenja AJAX poziva i SweetAlert-a

```
function delete_row(id = 0) {

    console.log('Row ID: ' + id);
    let url = null;

    <?php if($this->uri->segment_array()[2] === 'devices') :  ?>
    url = '<?php echo base_url(); ?>backend/devices/delete';

    <?php elseif($this->uri->segment_array()[2] === 'users') : ?>
    url = '<?php echo base_url(); ?>backend/users/delete';

    <?php elseif($this->uri->segment_array()[2] === 'groups') : ?>
    url = '<?php echo base_url(); ?>backend/groups/delete';

    <?php endif; ?>

    swal.fire({
        title: 'Are you Sure?',
        text: 'It will be delete permanently.',
        showCancelButton: true,
        confirmButtonColor: '#d33',
        cancelButtonColor: '#3CB371',
        cancelButtonText: 'Cancel',
        confirmButtonText: 'Yes! Delete it.',
        showLoaderOnConfirm: false
    }).then((result) => {
        console.log(result);
        if(!result.dismiss) {
            $.ajax({
                url: url,
                type: 'POST',
                data: {id: id},
                dataType: 'json',
                cache: false
            }).done(function(response) {
                console.log(response);
                if(response.status === 200) {
                    swal.fire({
                        type: 'success',
                        title: 'INFO',
                        html: response.message
                    });
                    $('tr[data-id="' + id + '"]').remove();
                }
                else {
                    swal.fire({
                        type: 'error',
                        title: 'ERROR',
                        html: response.message
                    })
                }
            });
        }
    });
}
```

Izvor 15: Autorica

U ovome je projektu prilikom uređivanja ikona, odnosno *font* elementa korišten FontAwesome alat (engl. *toolkit*) verzije 5.13.0.. FontAwesome je *toolkit* za *font* temeljen na CSS-u i LESS-u[2] (engl. *Leaner Style Sheets* – LESS). FontAwesome dolazi u dvije verzije; „*Free*" koja je besplatna i „*Pro*" koja se naplaćuje. (Fonticons Inc., https://fontawesome.com/, (28.08.2020.))

Osim gore navedenih biblioteka i alata za izradu *web*-aplikacije korišteni su i dodatni AdminLTE programski dodatci (engl. *plugin*); DataTables verzije 1.10.20 i SweetAlert2 verzije 9.10.8. DataTables je jQuery *plugin* koji olakšava izradu i pretraživanje HTML tablica. Ima ugrađene funkcionalnosti, kao što su primjerice paginacija, sortiranje i pretraga. (SpryMedia Ltd, https://datatables.net/manual/, (28.08.2020.))

Programski kod 3. Primjer inicijalizacije i definicije DataTable parametara

```
$('#device-table').DataTable({
    "paging": true,
    "lengthChange": true,
    "searching": true,
    "ordering": true,
    "info": true,
    "autoWidth": false,
    "responsive": true,
});
```

Izvor 16: Autorica

SweetAlert2 je JavaScript i CSS biblioteka koja zamjenjuje JavaScript *alert()* metode te omogućuje izradu skočnih prozora. (Monte et al., https://sweetalert2.github.io/, (28.08.2020.))

Za izradu grafova prikaza mrežnih podataka korištena je Smoothie Charts biblioteka verzije 1.36. Prilikom korištenja Smoothie Chart biblioteke definirana su dva objekta koja se

[2] *Less (which stands for Leaner Style Sheets) is a backwards-compatible language extension for CSS.* (Sellier et al., http://lesscss.org/, (30.08.2020.))

pune podatcima o trenutnom vremenu te podatcima o dolaznom ili odlaznom broju paketa prikupljenih pomoću socket-a.

Programski kod 4. Primjer korištenja SmoothieChart biblioteke

```
let txPackets    = new TimeSeries();
let rxPackets  = new TimeSeries();

socket.on('bandwidth_update', function(obj) {
    console.log(obj);
    txPackets.append(new Date().getTime(), testBytes(obj[0]['tx']));
    rxPackets.append(new Date().getTime(), testBytes(obj[0]['rx']));
});


let chart = new SmoothieChart({minValue:0, maxValue: 1000,
millisPerPixel:50, grid:{fillStyle:'#595959',strokeStyle:'#000'}});
chart.addTimeSeries(txPackets, { strokeStyle: 'rgba(0, 255, 0, 1)',
fillStyle: 'rgba(0, 255, 0, 0.2)', lineWidth: 4 });
chart.addTimeSeries(rxPackets, { strokeStyle:'rgb(255, 0, 0)',
fillStyle:'rgba(255, 0, 0, 0.3)', lineWidth:4 });
chart.streamTo(document.getElementById('bandwidth-chart'), 500);
```

Izvor 17: Autorica

6.2.MySQL

MySQL je SQL (engl. *Structured Query Language* - SQL) poslužitelj za upravljanje bazom podataka otvorenoga koda koji se koristi za *web*. Dakle, to je sustav za upravljanje bazom podataka koji se pokreće na poslužitelju. Pogodan je za izradu manjih i većih aplikacija, te koristi standardni SQL jezik za izradu upita. Moguće ga je kompajlirati na velikom broju platformi te je besplatan za korištenje. Podatci su pohranjeni u tablicama, odnosno kolekcijama povezanih podataka poredanih u stupce i redove. MySQL je jedna od najkorištenijih baza podataka otvorenoga koda općenito te najčešće korištena baza za izradu *web*-aplikacija. (Oracle Corporation, https://dev.mysql.com/doc/mysql-getting-started/en/, (28.08.2020.))

Za izradu ovoga projekta korišten je phpMyAdmin verzije 5.0.2, besplatan alat pisan u PHP programskom jeziku koji omogućuje administraciju MySQL baze putem *web*-a. PhpMyAdmin podržava veliki broj operacija, kao što su primjerice upravljanje tablicama,

relacijama, korisnicima, dozvolama, itd. putem korisničkog sučelja, ali isto tako omogućuje i direktno izvođenje SQL upita. (Bennetch, Bansod, Fauth, https://www.phpmyadmin.net/, (28.08.2020.))

6.3.NodeJS

Kao što je u poglavlju 6.1. prikazano, za izradu API dijela sustava za prikupljanje i prikaz mrežnih podataka korišten je NodeJS. NodeJS je asinkrono JavaScript poslužiteljsko okruženje otvorenoga koda koje izvršava JavaScript izvan *web*-preglednika. Jedna od glavnih odlika NodeJS-a jest asinkrona, *single-thread* komunikacija, što znači da se zahtjevi ne izvršavaju sinkrono, odnosno ne čeka se terminacija prethodnog zahtjeva za pokretanje idućeg zahtjeva, kao što je to primjerice slučaj sa PHP programskim jezikom,

NodeJS datoteke najčešće sadrže zadatke koji se pokreću određenim događajima (engl. *event*), kao što je primjerice pristup korisnika određenome *port*-u poslužitelja. NodeJS omogućuje instalaciju modula, odnosno skupinu funkcija koje želimo uključiti u projekt. Neki moduli su ugrađeni, što znači da se mogu koristiti bez prethodne instalacije. Jedan od ugrađenih modula je i HTTP modul (engl. *Hypertext Transfer Protocol* - HTTP), koji omogućuje prijenos podataka putem HTTP protokola, odnosno stvaranje HTTP poslužitelja koji prisluškuje definirane *port*-ove te vraća odgovore klijentima. (Refsnes, Refsnes, Refsnes, https://www.w3schools.com/nodejs/, (28.08.2020.))

6.3.1.Express

Za izradu poslužiteljskog dijela API-a korišten je Express okvir verzije 4.16.4. Express je NodeJS poslužiteljski okvir koji se postavlja pomoću Node Package Manager-a (NPM). (StrongLoop, IBM, https://expressjs.com/, (28.08.2020.))

Programski kod 5. Primjer uključenja Express modula u projekt

```
const express       = require('express');
const cors          = require('cors');
const http          = require('http');
const app           = express();
const server        = http.createServer(app);
const Chart         = require('./model/ChartModel.js');

const apiRoutes     = require('./routes/api');
const port          = 11337;
const host          = '0.0.0.0';

const httpd = app.listen(port, host, () => {
    console.log('Server Started Using IP: ' + host + ' - Using Port: ' +
port + ' - URL: http://' + host + ':' + port + '/status');
});

app.use(cors());
app.use('/', apiRoutes);
app.use(require('express-status-monitor')());
```

Izvor 18: Autorica

6.3.2.MySQL

Za uspostavu konekcije na bazu podataka u NodeJS-u, korišten je MySQL modul, nativni NodeJS modul pisan isključivo u JavaScript-u. (Wilson, fengmk2, Sidorov, https://github.com/mysqljs/mysql, (28.08.2020.))

Programski kod 6. Primjer kreiranja konekcije na bazu podataka

```
const mysql = require('mysql');

const connection = mysql.createConnection({
    host     : '127.0.0.1',
    user     : 'root',
    password : '',
    database : 'mispi_mikrotik'
});

module.exports = connection;
```

Izvor 19: Autorica

Kao i kod ostalih modula, prije korištenja, MySQL je potrebo uključiti funkcijom „*require*“. Zatim je definirana konekcija na bazu, koja je u ovome slučaju lokalna. Dakle, kreirana je konekcija na bazu podataka te su postavljeni podatci za pristup bazi.

Programski kod 7. Primjer SQL upita pomoću MySQL modula u NodeJS-u

```
const deviceSQL = 'SELECT mikrotik_devices.id, mikrotik_devices.username,
mikrotik_devices.password, mikrotik_devices.ip, mikrotik_devices.api as
port FROM mikrotik_devices WHERE mikrotik_devices.id = ?';
```

Izvor 20: Autorica

6.3.3.Routeros-client

Routeros-client je modul koji služi kao *wrapper* za Node-routeros API kojim se pojednostavljuje korištenje samog API-a, obzirom da je sama biblioteka „više“ objektno orijentirana u odnosu na Node-routeros biblioteku. Node-routeros API je jedna od temeljnih biblioteka ovoga projekta, radi se API-u koji je napisan u TypeScript-u isključivo za MikroTik RouterOS i NodeJS, a omogućuje radnje kao što su pokretanje i zaustavljanje *stream*-a, uspostavljanje konekcije te promjena poslužitelja, korisničkog imena i drugih parametara bez potrebe za ponovnim stvaranjem objekta. (Amaral, https://github.com/aluisiora/node-routeros/wiki, (28.08.2020.))

Programski kod 8. Primjer spajanja na bazu podataka i korištenje povratnih podataka za spajanje na MikroTik uređaj putem Routeros-client modula

```
Device.getIpDetails = function getIpDetails(device_id, param, param2,
result) {
    db.query(deviceSQL, device_id, function (error, data) {
        if(error) {
            result(error, null);
        }
        else {
            const credentials = new RouterOSClient({
                host: data[0]['ip'],
                port: data[0]['port'],
                user: data[0]['username'],
                password: data[0]['password'],
                keepalive: true
            });
            param = param.toLowerCase();
            console.log(param);
            credentials.connect().then((client) => {
                client.menu('/ip/address').where(param,
param2).getOnly().then((data) => {
                    result(null, data);
                }).catch((err) => {
                    result(null, err);
                });
            }).catch((err) => {
                result(null, err);
            });
        }
    });
}
```

Izvor 21: Autorica

Routeros-client služi za jednostavnije upravljanje operacijama, kao što su unosi, izmjene, brisanje i filtriranje podataka, koje API omogućuje. Svrha je olakšati pisanje koda korištenjem DRY metode razvoja (engl. *Don't Repeat Yourself* - DRY). Node-routeros je zapravo implementacija MikroTik API-a u NodeJS okruženju, odnosno JavaScript programskom jeziku. (Amaral, https://github.com/aluisiora/routeros-client/wiki, (28.08.2020.))

6.3.4.Socket.io

Za slanje *stream* podataka o dolaznom i odlaznom prometu pojedinih sučelja, odnosno parametara u Smoothie Chart varijable definirane u klijentskom dijelu sustava, korištena je socket.io biblioteka. Socket.io omogućuje dvosmjernu komunikaciju temeljenu na događajima

i u stvarnome vremenu između *web* preglednika i poslužitelja. On se sastoji od NodeJS poslužitelja i JavaScript klijentske biblioteke za *web* preglednik. Socket.io koristi WebSocket komunikacijski protokol koji uspostavlja *full-duplex* bidirekcijski kanal između klijenta i poslužitelja, dakle podatci mogu teći istovremeno u oba smjera. (Arrachequesne, Little, https://socket.io/docs/, (28.08.2020.))

Programski kod 9. : Primjer inicijalizacije socket-a i slušanja konekcije

```
const io = require('socket.io')(httpd);
app.set('socketio', io);

io.on('connection', function(socket) {
    console.log('Connected successfully to the socket ...');
    socket.on('device_data', function(params) {
        console.log(params);
        console.log(params['device_id']);
        Chart.getChartData2(params['device_id'], params['interface'],
socket);
    });
    socket.on('disconnect', function() {
        console.log('user disconnected');
    });
}).setMaxListeners(0);
```

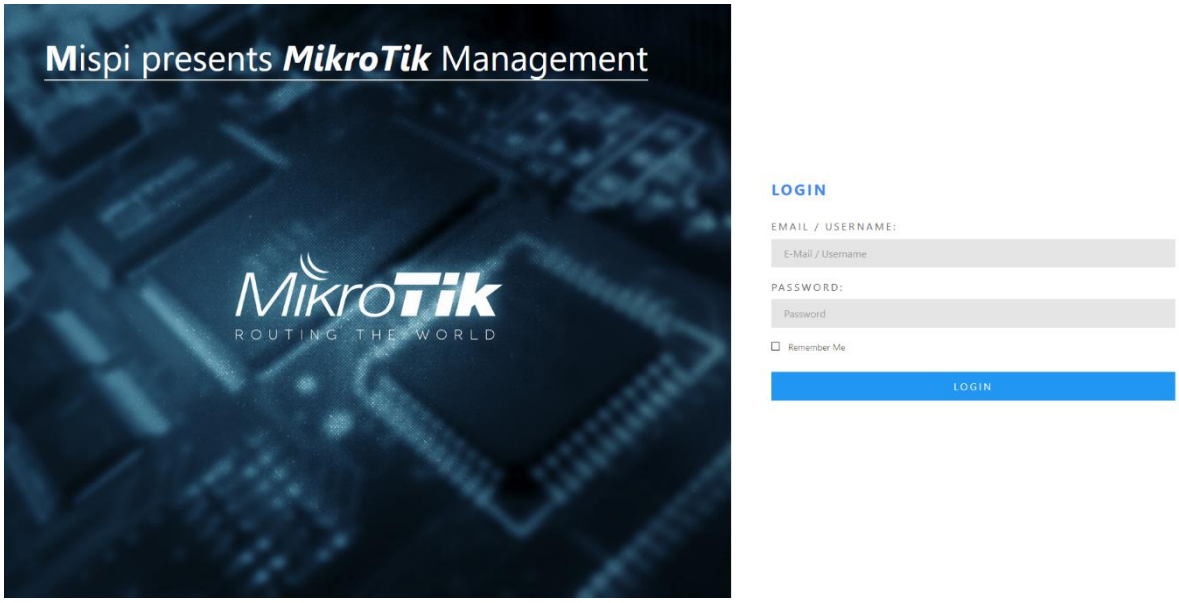
Izvor 22: Autorica

7. PREGLED KLIJENTSKOG DIJELA RJEŠENJA

U ovome poglavlju prikazati će se konačan izgled klijentskog dijela razvijenog sustava za prikupljanje i pregled mrežnih podataka sa MikroTik uređaja kroz niz prikaza pregleda zaslona korisnika *web*-aplikacije.

Na slici 13. je prikazan zaslon koji se javlja prilikom autentifikacije u sustav. Korisnik unosi korisničke podatke te se prijavljuje u sustav.

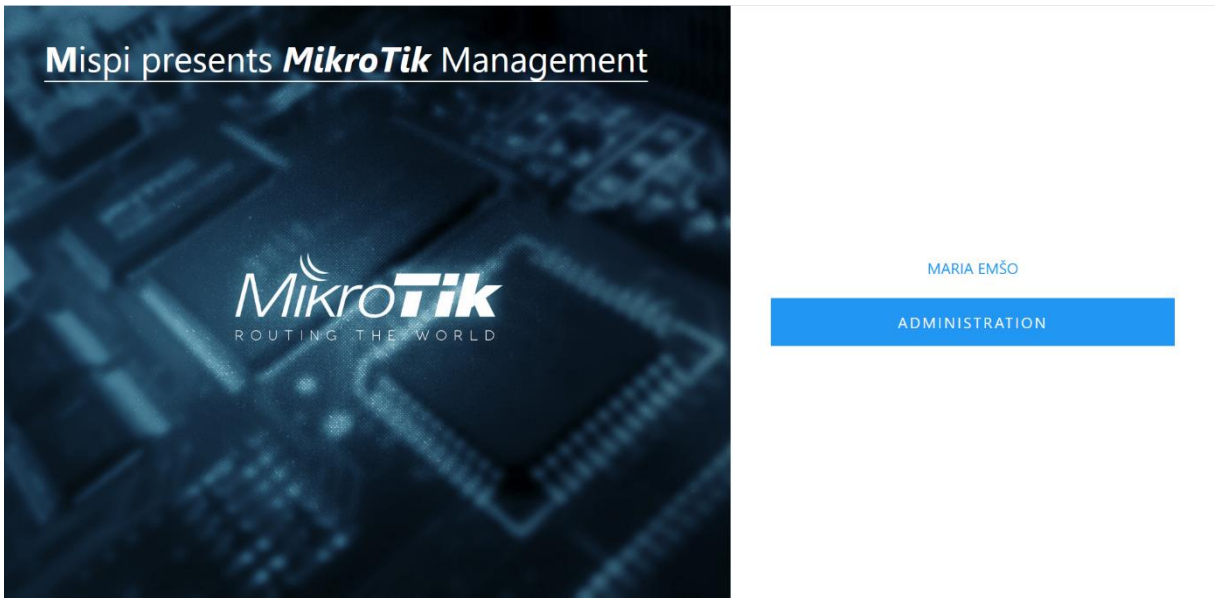
Slika 13. Pregled autentifikacije u sustav



Izvor 23: Autorica

Na slici 14. prikazana je početna stranica koja se javlja korisniku *web*-aplikacije nakon postupka autentifikacije u sustav. Korisnik odabire „*Administration*“ čime mu se otvara kontrolna ploča sa izbornom trakom.

Slika 14. Pregled početne stranice prikaza sustava

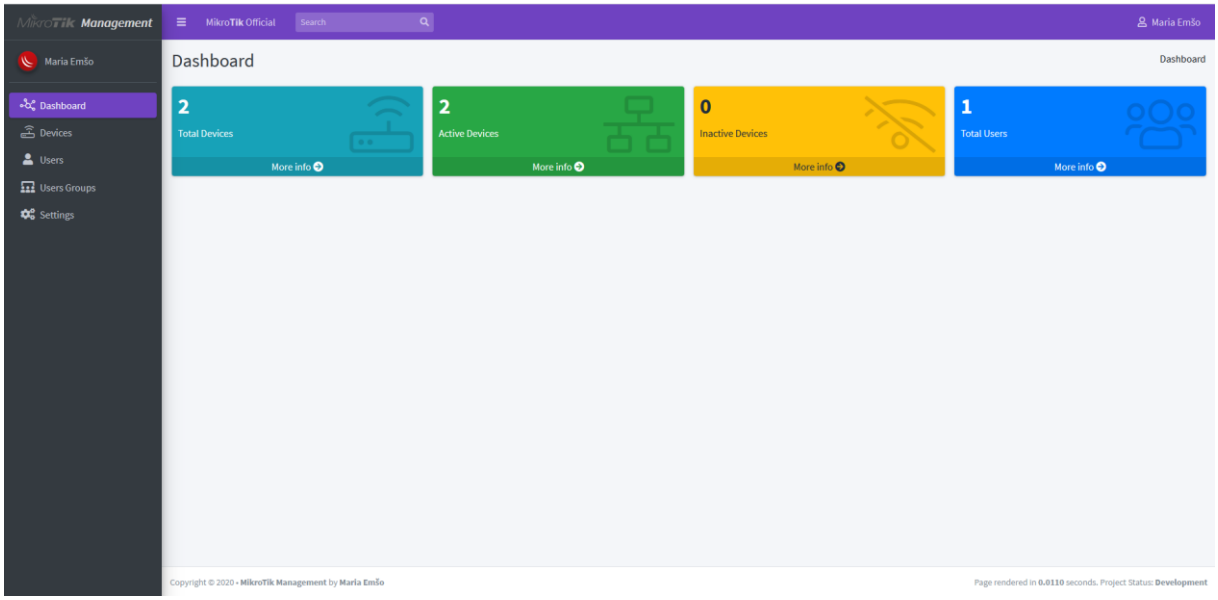


Izvor 24: Autorica

Na slici 15. je prikazan zaslon kontrolne ploče koji se javlja korisniku *web*-aplikacije nakon odabira administracije. Korisniku su vidljivi podatci o svim uređajima koji se trenutno nalaze u nadzoru, zajedno sa podatcima o ukupnom broju korisnika sustava. Sa lijeve strane nalazi se izborna traka koja se može smanjiti ili proširiti dok se u gornjem desnom kutu nalaze podatci o prijavljenom korisniku. Klikom na korisnika otvara se izbornik koji omogućuje odjavu iz sustava ili pregled podataka o korisničkom profilu.

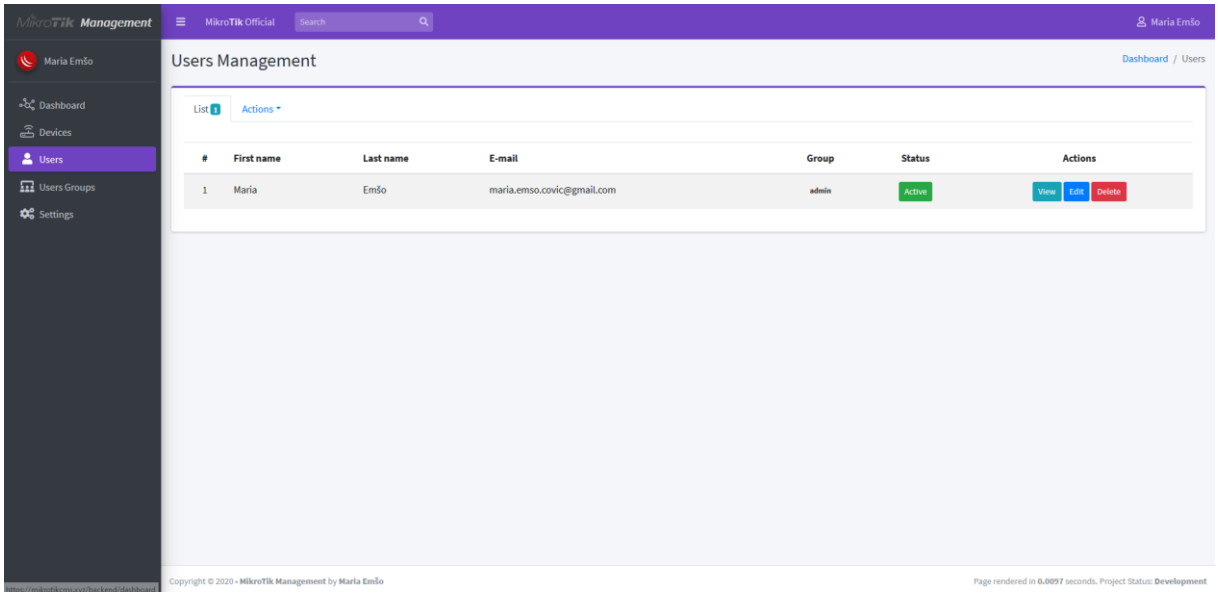
Na slici 16. prikazan je pregled korisnika sustava. Vidljivi su podatci o imenu, prezimenu i adresi e-pošte korisnika. Korisnički podatci se mogu pregledati, izmijeniti ili obrisati dok se klikom na padajući izbornik „*Actions*“ otvara mogućnost odabira između dodavanja novog korisnika u sustav ili izvoza podataka o postojećim korisnicima sustava.

Slika 15. Pregled kontrolne ploče sustava



Izvor 25: Autorica

Slika 16. Pregled korisnika sustava

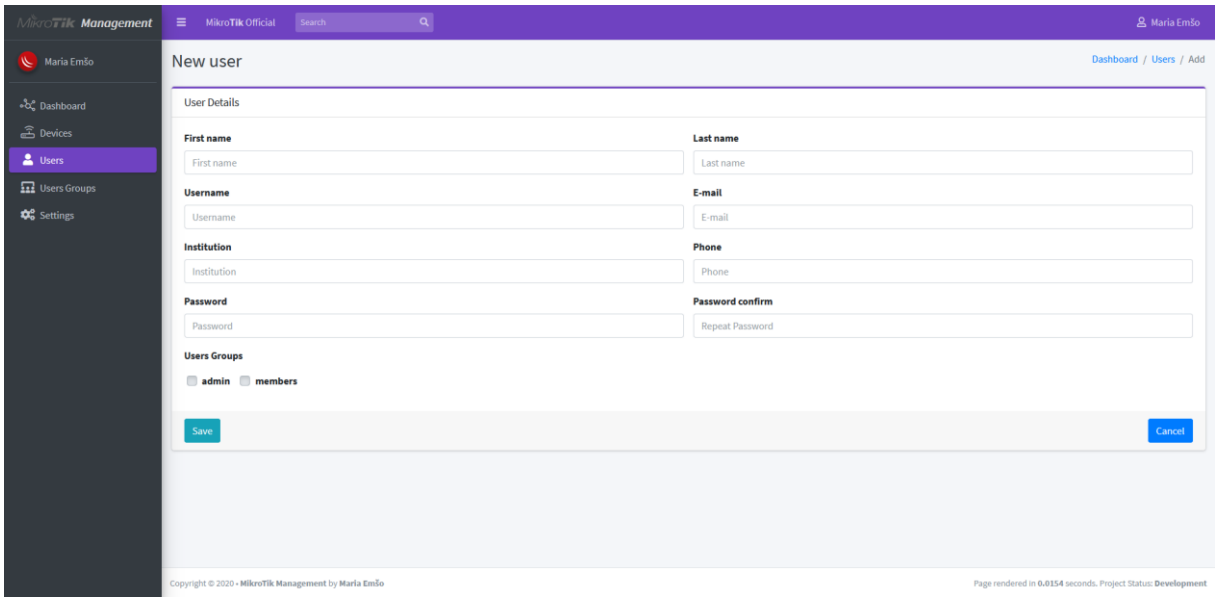


Izvor 26: Autorica

Na slici 17. prikazan je pregled forme za unos novog korisnika u sustav koja se otvara nakon odabira unosa novog korisnika u zaslonu pregleda korisnika. Prilikom unosa korisnika

potrebno je ispuniti podatke o imenu, prezimenu, korisničkom imenu, adresi e-pošte, instituciji i kontakt telefonu. Potrebno je i odabrati pripadnost korisničkoj grupi te unesti lozinku u formu validacije.

Slika 17. Pregled forme za unos korisnika u sustav

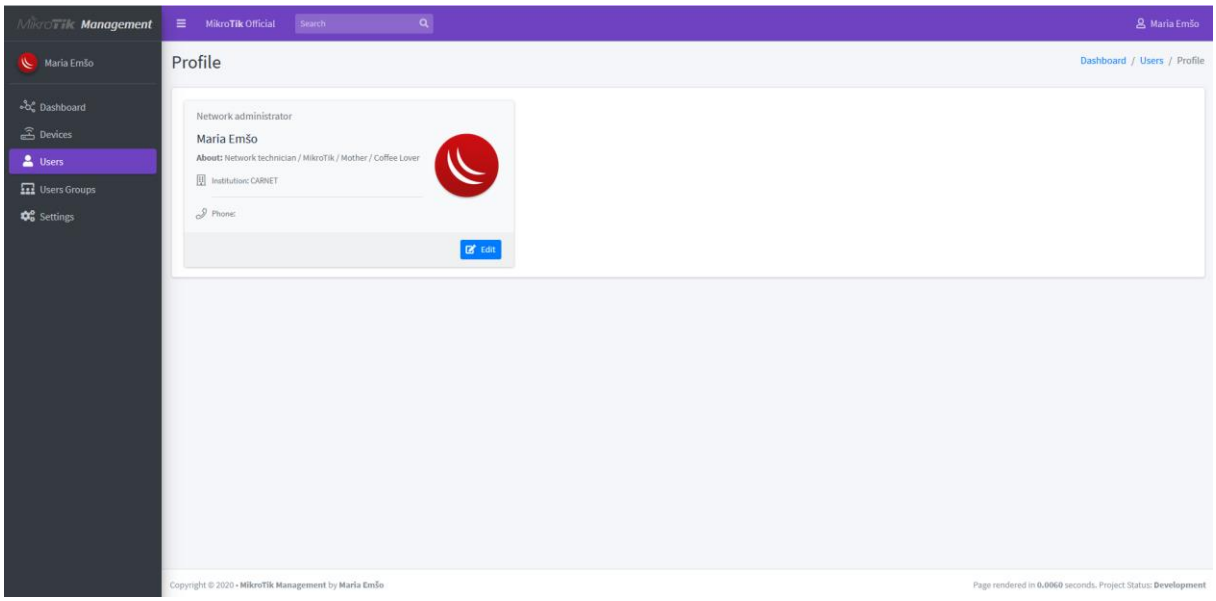


Izvor 27: Autorica

Na slici 18. prikazan je zaslon pregleda detalja korisnika, odnosno pregled profila. Iz pregleda profila, klikom na gumb „*Edit*“ moguće je pristupiti formi za izmjenu podataka o korisniku.

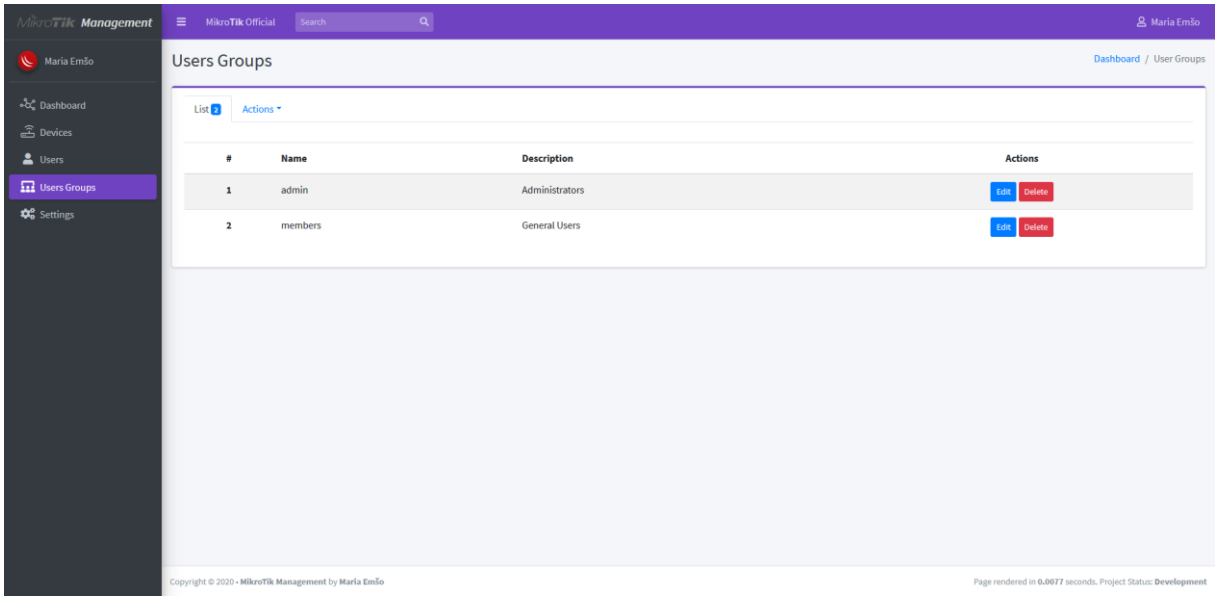
Na slici 19. prikazan je zaslon pregled korisničkih grupa. Korisničke grupe se mogu izmijeniti ili obrisati dok se klikom na padajući izbornik „Actions“ otvara mogućnost unosa nove korisničke grupe ili izvoza podataka o postojećim korisničkim grupama.

Slika 18. Pregled korisničkog profila



Izvor 28: Autorica

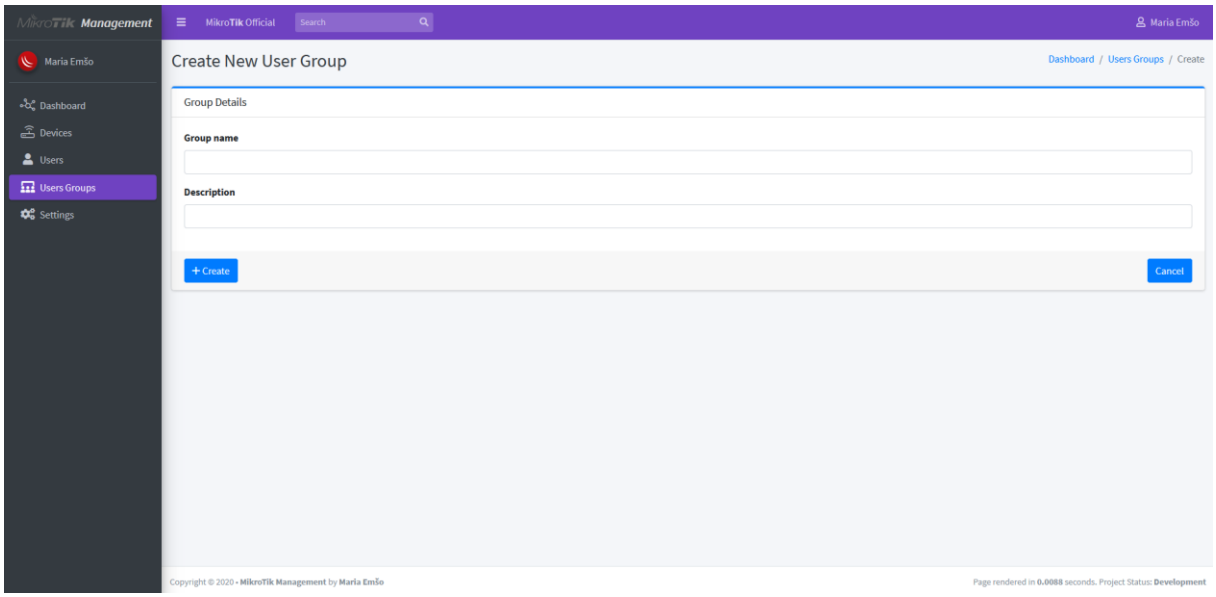
Slika 19. Pregled korisničkih grupa



Izvor 29: Autorica

Na slici 20. prikazan je zaslon unosa nove korisničke grupe koji se otvara nakon odabira unosa nove korisničke grupe u zaslonu pregleda korisničkih grupa. Prilikom unosa nove korisničke grupe unose se podatci o nazivu grupe te opis grupe.

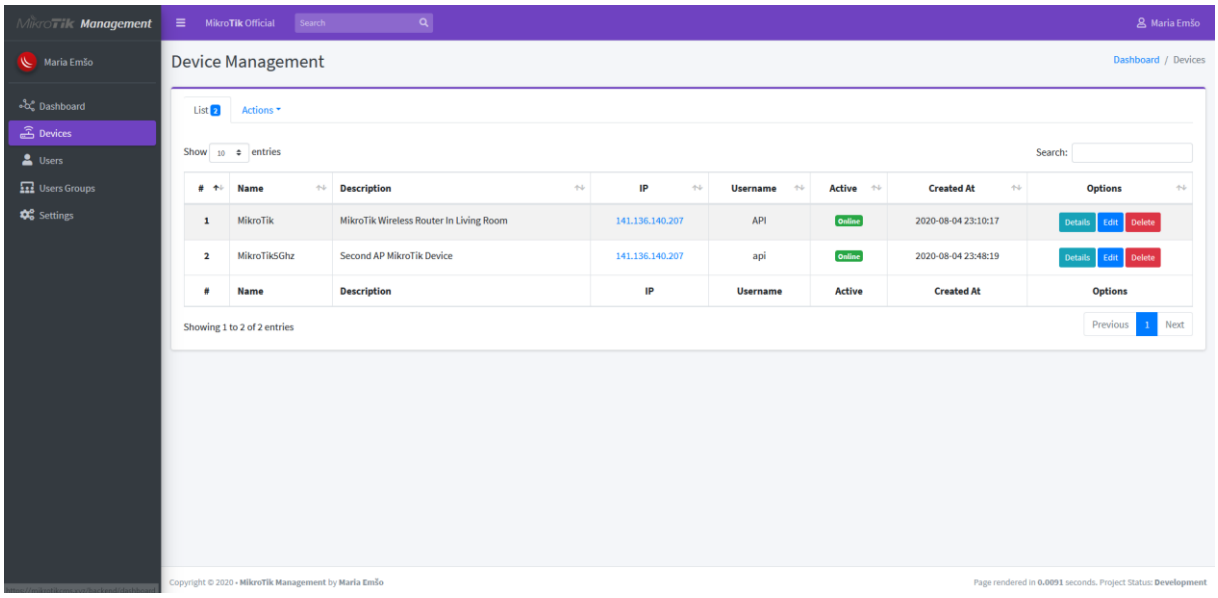
Slika 20. Pregled unosa korisničke grupe u sustav



Izvor 30: Autorica

Na slici 21. prikazan je zaslon pregleda uređaja koji se nalaze u nadzornom sustavu. Prikazuju se podatci o nazivu, opisu, javnoj IP adresi i uređaja te datum kada je uređaja unesen u nadzor. Uređaji se mogu dodatno pregledati, izmijeniti ili obrisati, dok se klikom na padajući izbornik „*Actions*“ otvara mogućnost unosa novog uređaja u nadzor ili izvoza podataka o postojećim uređajima.

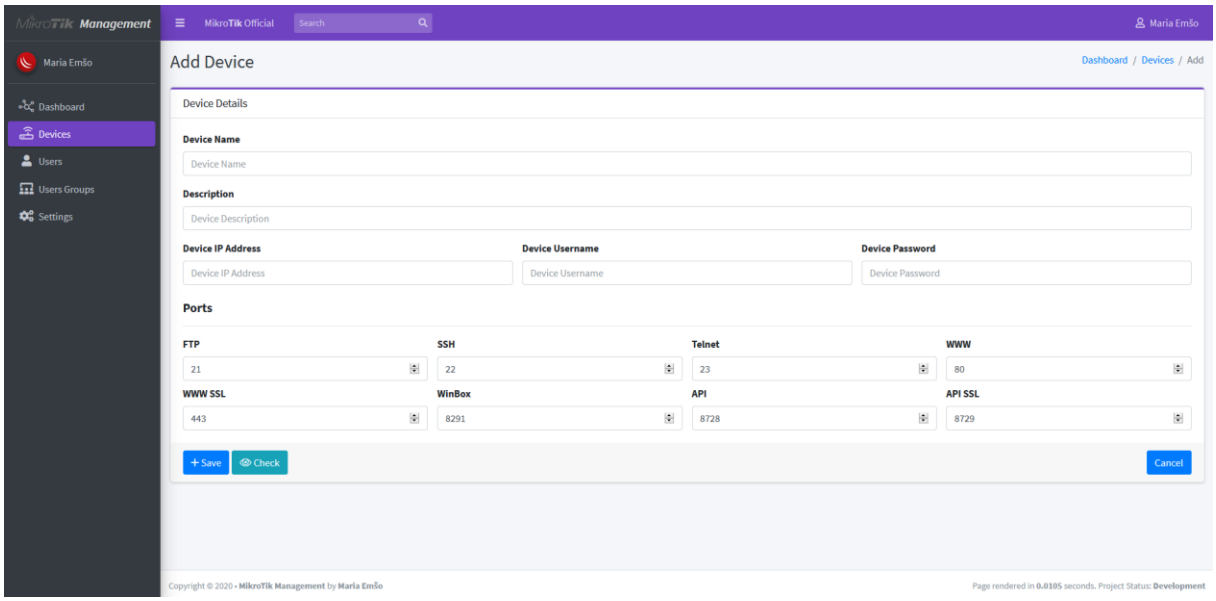
Slika 21. Pregled uređaja



Izvor 31: Autorica

Na slici 22. prikazan je zaslon unosa novog uređaja koji se otvara nakon odabira unosa novog uređaja u zaslonu pregleda uređaja. Prilikom unosa novog uređaja upisuju se podatci o nazivu, opisu, IP adresi te korisničkom imenu i lozinki koji su postavljeni za pristup API servisa na MikroTik mrežnom uređaju. Također se unose podatci o podešenim *port*-ovima, odnosno unutar forme za izmjenu podataka o otvorenim *port*-ovima po servisima moguće je izmijeniti zadane postavke *port*-ova, te u slučaju potrebe, izmijeniti API *port* za pristup servisa ovisno o postavkama MikroTik mrežnog uređaja. Nakon unosa podataka o IP adresi, korisničkom imenu i lozinki te API *port*-u koji se koristi za spajanje, može se, klikom na gumb „*Check*“ pokrenuti provjera konekcije. U slučaju da je konekcija uspostavljena, javlja se obavijest sa podatcima o uređaju, u suprotnom javlja se obavijest isteka vremena.

Slika 22. Pregled unosa uređaja u sustav



Izvor 32: Autorica

Na slici 23. prikazan je zaslon pregleda podataka o uređaju kojemu se pristupa odabirom pregleda detalja o uređaju na zaslonu pregleda svih uređaja. Ovdje se prikazuju detaljni podatci o resursima odabranog MikroTik uređaja, odnosno podatci o iskorištenosti memorije i procesora, vremenu aktivnog rada, serijskom broju i modelu samog uređaja te trenutnom i nadogradivom ROS-u.

Prikazuju se i podatci o samoj konfiguraciji uređaja, odnosno postavljenim logičkim sučeljima, definiranim adresnim rasponima te fizičkim sučeljima. Za postavljena logička sučelja prikazuju se podatci o nazivu i tipu sučelja, postavljeni MTU (engl. *Maximum Transmission Unit* - MTU), MAC adresa (engl. *Media Access Control* - MAC) sučelja te podatci o samom statusu i aktivnosti sučelja, odnosno vrijeme aktivnosti, trenutna aktivnost i status. Za definirane adresne raspone prikazuju se podatci o mreži i IP adresnom rasponu, sučelju kojem je adresni raspon dodijeljen te status samog raspona. Za fizička sučelja prikazuju se podatci o nazivima, dodijeljenim logičkim sučeljima, uključenju hardverskog usmjeravanja te samom statusu sučelja. Osim podataka o konfiguraciji uređaja, tablično se prikazuju i podatci

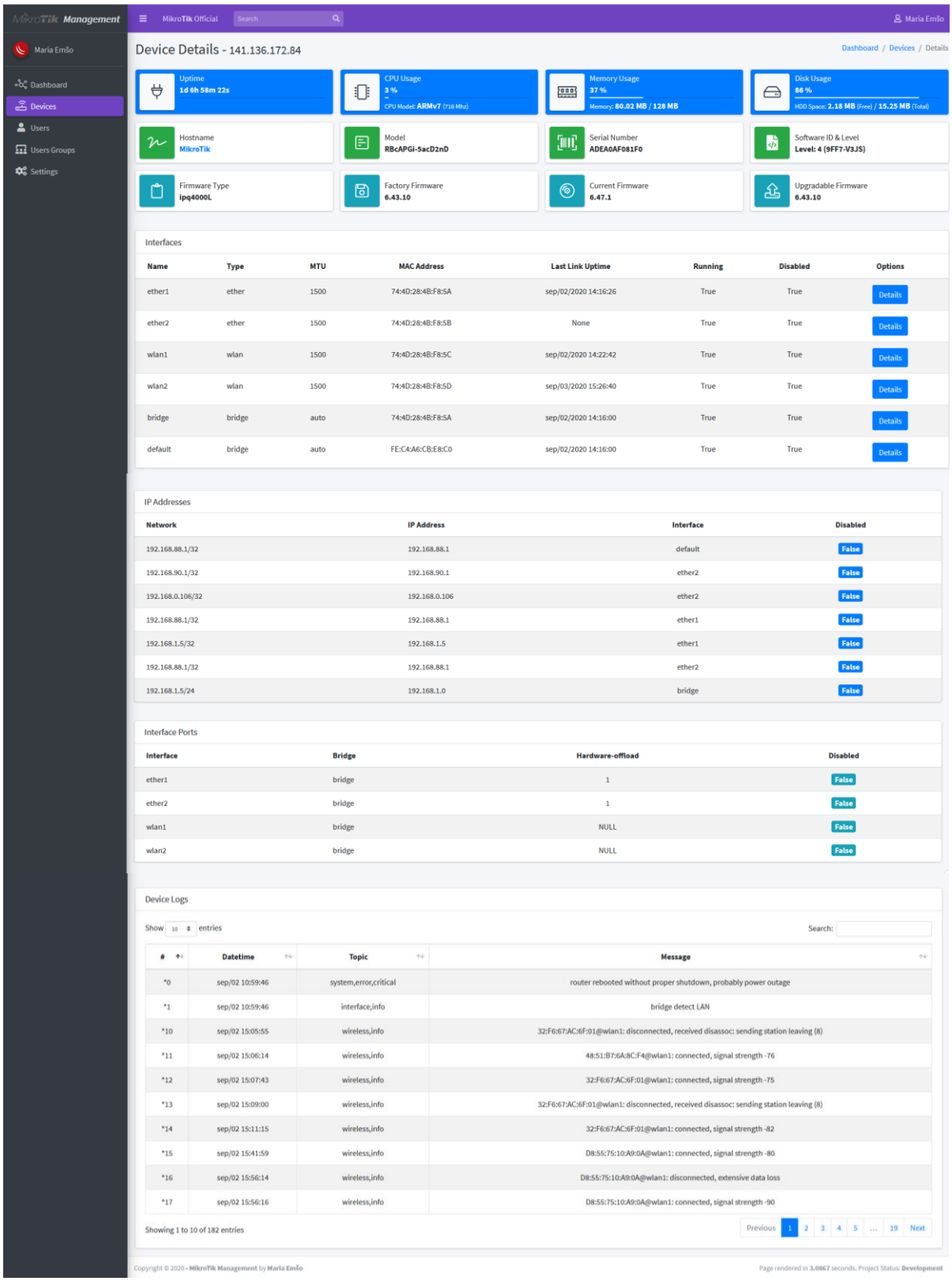
sa zapisima koje je prikupio uređaj. Za zapise se prikazuju podatci o datumu i vremenu kada su zabilježeni, tipu zapisa te samoj poruci zapisa. Zapisi se mogu izlistavati, sortirati i pretraživati.

Na slici 23. prikazan je zaslon pregleda podataka o uređaju kojemu se pristupa odabirom pregleda detalja o uređaju na zaslonu pregleda svih uređaja. Ovdje se prikazuju detaljni podatci o resursima odabranog MikroTik uređaja, odnosno podatci o iskorištenosti memorije i procesora, vremenu aktivnog rada, serijskom broju i modelu samog uređaja te trenutnom i nadogradivom ROS-u.

Na slici 24. prikazan je zaslon pregleda grafova pometa sučelja, kojemu se pristupa klikom na gumb „*Details*“ na zaslonu pregleda odabranog uređaja unutar pregleda podataka o logičkim sučeljima. Graf prikazuje trenutnu *download* i *upload* potrošnju odabranog sučelja u Mbps (engl. *Megabit per second* - Mbps). *Download* potrošnja označena je crvenom bojom, a *upload* zelenom. Osim grafa potrošnje prikazuju se i detalji o postavkama samog sučelja.

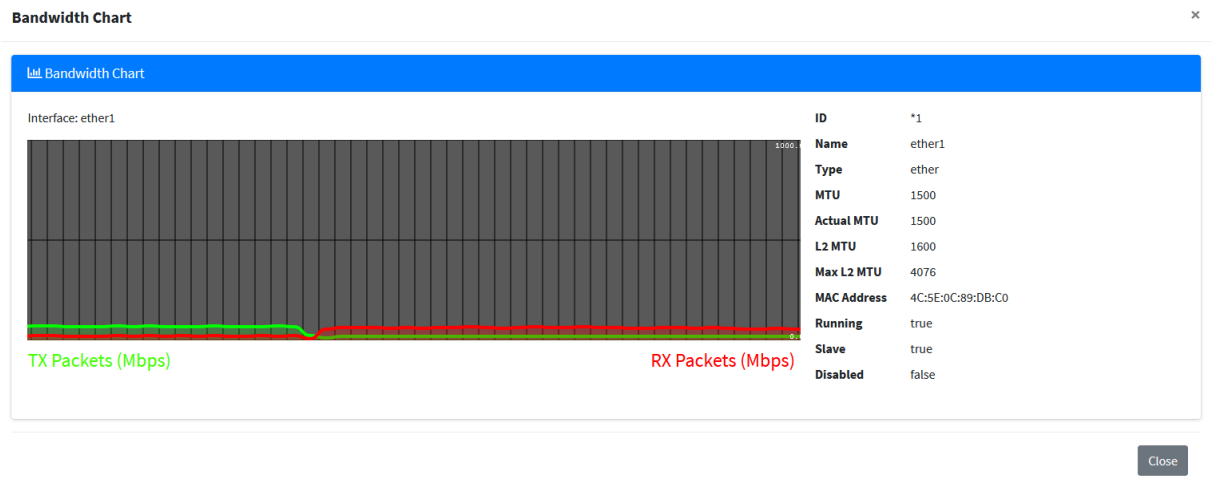
Na slici 25. prikazan je pregled zaslona postavki sustava. Ovdje se prikazuju podatci o postavkama sustava za nadzor mreže, odnosno podatci o korištenoj NodeJS verziji, poslužitelju, PHP verziji, bazi podataka i sustavu za upravljanje bazom podataka te raspoloživim i iskorištenim memorijskim resursima poslužitelja. Osim pregleda podataka omogućena je i izmjena podataka o NodeJS poslužitelju putem forme za izmjenu. Izmijeniti se mogu podatci o IP adresi ili *hostname*-u poslužitelja te *port*-u putem kojeg vrši komunikacija.

Slika 23. Pregled mrežnih podataka uređaja



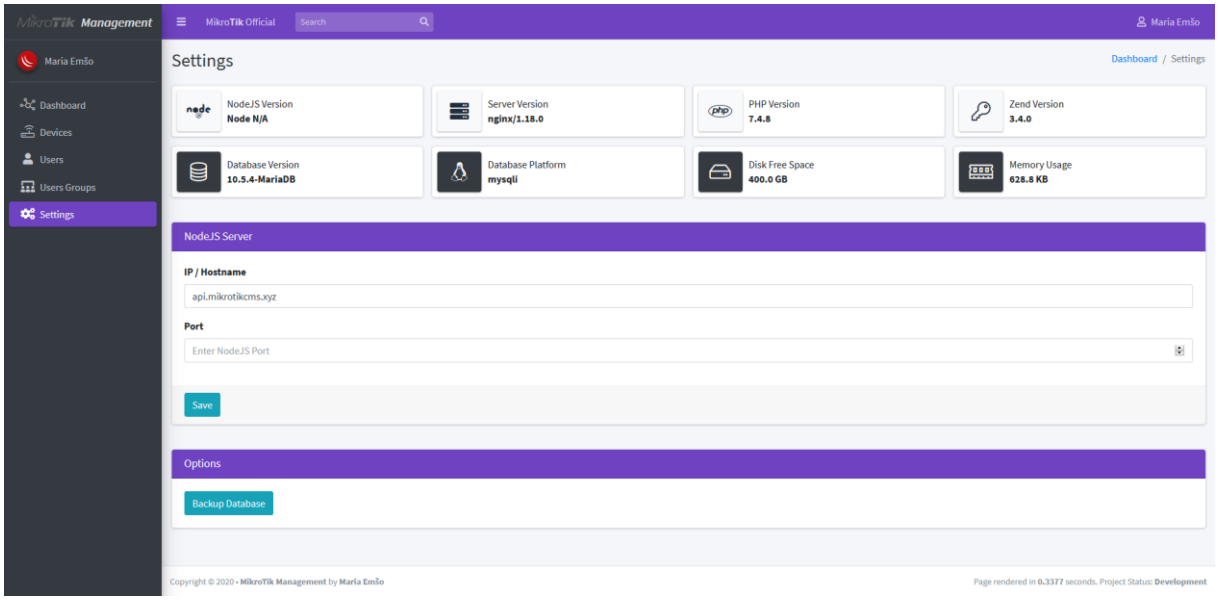
Izvor 33: Autorica

Slika 24. Pregled grafičkog prikaza mrežnih podataka sučelja uređaja



Izvor 34: Autorica

Slika 25. Pregled postavki sustava



Izvor 35: Autorica

8. DALJNJI RAZVOJ

Razvijeni sustav za prikupljanje i prikaz mrežnih podataka zadovoljava većinu početno specificiranih zahtjeva. „*Must have*“ zahtjevi, odnosno; prikupljanje mrežnih podataka sa uređaja, prikaz prikupljenih mrežnih podataka, unos, izmjena i brisanje mrežnih podataka iz nadzora te autentifikacija u sustav, su omogućeni. Ispunjeni su i početno postavljeni „*Should have*“ zahtjevi, odnosno; grafički prikaz prikupljenih podataka te upravljanje korisnicima sustava. Djelomično su ispunjeni i neki od „*Could have*“ zahtjeva, odnosno; mogućnost izvoza podataka te djelomično upravljanje postavkama, kroz mogućnost izmjene podataka o NodeJS poslužitelju. Daljnji razvoj sustava mogao bi se temeljiti na razvoju preostalih, djelomično ispunjenih „*Should have*“ zahtjeva, odnosno na dodavanju mogućnosti uvoza podataka u sustav, unaprjeđenje upravljanja postavkama sustava, kao što je primjerice personalizacija izgleda kontrolne ploče, te dodavanje mogućnosti spremanja zapisa o izvršenim korisničkim izmjenama kroz nadzorni sustav u bazu podataka. Sustav bi se moglo unaprijediti i kroz razvoj mogućnosti koje nisu predstavljene u početnim zahtjevima, kao što je to primjerice definiranje uloga i ograničenja ovisno o pripadnosti određenim korisničkim grupama. Ukoliko bi se sustav implementirao u produkciju valjalo bi unaprijediti prikaz mrežnih podataka, odnosno proširiti količinu podataka koja se izvlači sa uređaja te dodatno kategorizirati pregled istih uz proširenje same izborne trake pregleda uređaja.

Razvijeni sustav za prikupljanje i prikaz mrežnih podataka, kao što je u početku definirano, nije ispunio „*Won't have*“ funkcionalnosti, odnosno ne omogućuje rekonfiguraciju i direktno upravljanje mrežnim postavkama uređaja. Kao daljnji razvoj, navedene funkcionalnosti bi se mogle implementirati uz pomoć definiranja ograničenja korisničkih grupa, čime bi se sustav dodatno osigurao a omogućile bi se neke od najjednostavnijih radnji, kao što je to primjerice *restart* uređaja ili nadogradnja ROS-a.

9. ZAKLJUČAK

U ovome je radu razvijen sustav za prikupljanje i prikaz mrežnih podataka sa MikroTik mrežnih uređaja. Tema rada nastala je kao posljedica autoričina rada u jednom mrežnom odjelu, gdje aktivno radi sa MikroTik mrežnim uređajima, te utjecaja projekta poslodavca koji uključuje izradu dokumentacije programskog rješenja koje će se koristiti za nadzor buduće lokalne mreže korisnika kojima su pružene usluge Internet-a.

Prije izrade samog rješenja, napravljena je razrada sustava, odnosno definirani su zahtjevi koje sustav mora zadovoljiti. Zahtjevi su predstavljeni pomoću MoSCoW metode analize. Zatim su definirane pojedine funkcionalnosti putem dijagrama slučaja korištenja, dok su pojedini slučajevi korištenja razrađeni u dijagrame aktivnosti. Nakon specifikacije zahtjeva i funkcionalnosti, predočena je sustavna, mrežna i programska arhitektura razvoja te je izrađen relacijski model podataka na temelju kojeg je izgrađena baza podataka. Za razvoj sustava korištene su različite tehnologije i programski jezici, poput PHP-a, JavaScript-a, CSS-a, AJAX-a, HTML-a i SQL-a. Korišteni su i različiti programski dodatci, okviri i moduli, primjerice jQuery, Bootstrap, AdminLTE, Smoothie Charts, Express, Socket.io, Routeros-client, itd.

Stvoreni sustav objedinjuje mrežne i programske zahtjeve u cjelinu koja omogućuje nadzor mreže. Sustav se sastoji od dviju temeljnih komponenti; API koji je razvijen uz pomoć NodeJS-a, i koji komunicira sa MikroTik mrežnim uređajem i dohvaća mrežne podatke, te *web*-aplikacija koja je razvijena pomoću CodeIgniter programskog okvira i koja omogućuje korisnicima sustava pregled prikupljenih podataka te upravljanje samim sustavom putem *web* GUI-a (engl. *Graphical User Interface* - GUI).

Izrađeni sustav je još uvijek u razvoju te ga je potrebno unaprijediti dodatnim mogućnostima kao što je spremanje korisničkih zapisa, definiranje uloga i ograničenja, proširenje i unaprjeđenje panela pregleda podataka mrežnih uređaja te mogućnost upravljanja i izmjene konfiguracije na samim uređajima.

POPIS KORIŠTENIH KRATICA

ROS - Router Operating System
IP - Internet Protocol address
ISP - Internet Service Provider
CLI - Command Line Interface
API - Application Programmable Interface
MoSCoW - Must have, Should havem Could have, Won't have
UML - Unified Modeling Language
MVC - Model-view-controller
DSL - Digital Subscriber Line
PHP - Hypertext Preprocessor
CGI - Common Gateway Interface
HTML - HyperText Markup Language
itd. – i tako dalje
RAD - Rapid Application Development
CSS - Cascading Style Sheets
UI - User Interface
AJAX - Asynchronous JavaScript and Extensible Markup Language
DOM - Document Object Modeling
LESS - Leaner Style Sheets
SQL - Structured Query Language
HTTP – Hypertext Transfer Protocol
DRY - Don't Repeat Yourself
MTU - Maximum Transmission Unit
MAC - Media Access Control
Mbps - Megabit per second
GUI - Graphical User Interface

POPIS LITERATURE

1. Achour, M., et al, PHP manual, https://www.php.net/manual/en/, (28.08.2020.)
2. Amaral, A., Node RouterOS, https://github.com/aluisiora/node-routeros/wiki, (28.08.2020.)
3. Amaral, A., RouterOS Client, https://github.com/aluisiora/routeros-client/wiki, (28.08.2020.)
4. Arrachequesne, D., Little E., Socket.io, https://socket.io/docs/, (28.08.2020.)
5. Bennetch, I., Bansod, D., Fauth, M. M., phpMyAdmin, https://www.phpmyadmin.net/, (28.08.2020.)
6. CodeIgniter Foundation, CodeIgniter, https://codeigniter4.github.io/userguide/intro/index.html, (28.08.2020.)
7. Colorlib, AdminLTE , https://github.com/ColorlibHQ/AdminLTE, (28.08.2020.)
8. Colorlib, AdminLTE Docs, https://adminlte.io/docs/3.0/dependencies.html, (28.08.2020.)
9. Fonticons Inc., Font Awesome, https://fontawesome.com/, (28.08.2020.)
10. Monte, L., et al., SweetAlert2, https://sweetalert2.github.io/, (28.08.2020.)
11. Oracle Corporation, Getting Started with MySQL, https://dev.mysql.com/doc/mysql-getting-started/en/, (28.08.2020.)
12. Otto M., et al., Bootstrap, https://getbootstrap.com/docs/4.5/getting-started/introduction/, (28.08.2020.)
13. Refsnes, H., Refsnes, S., Refsnes, J. E., Node.js Tutorial, https://www.w3schools.com/nodejs/, (28.08.2020.)
14. Sellier,A., et al., Less.js, http://lesscss.org/, (30.08.2020.)
15. SIA Mikrotīkls, About us, https://mikrotik.com/aboutus, (20.08.2020.)
16. SIA Mikrotīkls, Manual:API, https://wiki.mikrotik.com/wiki/Manual:API, (20.08.2020)
17. SIA Mikrotīkls, Manual:RouterOS features, https://wiki.mikrotik.com/wiki/Manual:RouterOS_features, (20.08.2020)
18. SIA Mikrotīkls, Manual:Webfig, https://wiki.mikrotik.com/wiki/Manual:Webfig (28.08.2020.)
19. SpryMedia Ltd, DataTables Manual, https://datatables.net/manual/, (28.08.2020.)
20. StrongLoop, IBM, Express.js, https://expressjs.com/, (28.08.2020.)
21. The jQuery Foundation, jQuery, https://jquery.com/, (28.08.2020.)

22. Walnes, J., Noakes, D., Smoothie Charts, http://smoothiecharts.org/ (28.08.2020.)
23. Wilson, D., fengmk2, Sidorov A., MySQL.js, https://github.com/mysqljs/mysql, (28.08.2020.)
24. Zivolo, F., Popper.js, https://popper.js.org/docs/v2/, (28.08.2020.)

POPIS TABLICA

Tablica 1. MoSCoW metoda analize zahtjeva..... 6

POPIS FOTOGRAFIJA

Slika 1. MikroTik 3
Slika 2. Dijagram slučaja korištenja 8
Slika 3. Dijagram aktivnosti – „Autentifikacija“ 9
Slika 4. Dijagram aktivnosti – „Dashboard“ 10
Slika 5. Dijagram aktivnosti – „Users“ 12
Slika 6. Dijagram aktivnosti – „Users Groups“ 14
Slika 7. Dijagram aktivnosti – „Devices“ 16
Slika 8. Dijagram aktivnosti – „Settings“ 17
Slika 9. Shematski prikaz arhitekture sustava 19
Slika 10. Shematski prikaz mrežne arhitekture 20
Slika 11. Shematski prikaz programske arhitekture 21
Slika 12. Relacijski model baze podataka 23
Slika 13. Pregled autentifikacije u sustav 35
Slika 14. Pregled početne stranice prikaza sustava 36
Slika 15. Pregled kontrolne ploče sustava 37
Slika 16. Pregled korisnika sustava 37
Slika 17. Pregled forme za unos korisnika u sustav 38
Slika 18. Pregled korisničkog profila 39
Slika 19. Pregled korisničkih grupa 39
Slika 20. Pregled unosa korisničke grupe u sustav 40
Slika 21. Pregled uređaja 41
Slika 22. Pregled unosa uređaja u sustav 42
Slika 23. Pregled mrežnih podataka uređaja 44
Slika 24. Pregled grafičkog prikaza mrežnih podataka sučelja uređaja 45
Slika 25. Pregled postavki sustava 45

POPIS PROGRAMSKOG KODA

Programski kod 1. Primjer korištenja CodeIgniter insert funkcije 25
Programski kod 2. Primjer korištenja AJAX poziva i SweetAlert-a 27
Programski kod 3. Primjer inicijalizacije i definicije DataTable parametara 28
Programski kod 4. Primjer korištenja SmoothieChart biblioteke 29
Programski kod 5. Primjer uključenja Express modula u projekt 31
Programski kod 6. Primjer kreiranja konekcije na bazu podataka 31
Programski kod 7. Primjer SQL upita pomoću MySQL modula u NodeJS-u 32
Programski kod 8. Primjer spajanja na bazu podataka i korištenje povratnih podataka za spajanje na MikroTik uređaj putem Routeros-client modula 33
Programski kod 9. : Primjer inicijalizacije socket-a i slušanja konekcije 34