



УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

НОВИ САД

Департман за рачунарство и аутоматику

Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: Немања Лазукић

Број индекса: РА 228/2014

Тема рада: Имплементација система за руковање претплатницима и садржајем у дигиталној телевизији

Ментор рада: проф. др Илија Башичевић

Нови Сад, мај, 2021



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

| | | | |
|---|--|----------------------------|----------------|
| Редни број, РБР: | | | |
| Идентификациони број, ИБР: | | | |
| Тип документације, ТД: | Монографска документација | | |
| Тип записа, ТЗ: | Текстуални штампани материјал | | |
| Врста рада, ВР: | Завршни (Bachelor) рад | | |
| Аутор, АУ: | Немања Лазукић | | |
| Ментор, МН: | проф. др Илија Башичевић | | |
| Наслов рада, НР: | Имплементација система за руковање претплатницима и садржајем у дигиталној телевизији | | |
| Језик публикације, ЈП: | Српски / латиница | | |
| Језик извода, ЈИ: | Српски | | |
| Земља публиковања, ЗП: | Република Србија | | |
| Уже географско подручје, УГП: | Војводина | | |
| Година, ГО: | 2021. | | |
| Издавач, ИЗ: | Ауторски репринт | | |
| Место и адреса, МА: | Нови Сад; трг Доситеја Обрадовића 6 | | |
| Физички опис рада, ФО: (поглавља/страница/цитата/табела/слика/графика/прилога) | 7/49/6/23/26/0/0 | | |
| Научна област, НО: | Електротехника и рачунарство | | |
| Научна дисциплина, НД: | Рачунарска техника | | |
| Предметна одредница/Кључне речи, ПО: | Дигитална телевизија, Рест-АПИ, ХТТП протокол, База података | | |
| УДК | | | |
| Чува се, ЧУ: | У библиотеци Факултета техничких наука, Нови Сад | | |
| Важна напомена, ВН: | | | |
| Извод, ИЗ: | У овом раду описано је једно решење, система за управљање корисницима, уређајима и садржајем. Систем је састављен из двије главне целине, фронтенда и бекенда. Фронтенд је одговоран за слање захтјева ка бекенду, примање одговора и приказ добављених података. Бекенд је одговоран за обраду захтјева, упис и испис из базе података и враћање одговора ка фронтенду. | | |
| Датум прихваташа теме, ДП: | | | |
| Датум одбране, ДО: | | | |
| Чланови комисије, КО: | Председник: | проф. др Мирослав Поповић | |
| | Члан: | проф. др Небојша Пјевалица | Потпис ментора |
| | Члан, ментор: | проф. др Илија Башичевић | |



KEY WORDS DOCUMENTATION

| | | | |
|---|--|------------------------|----------------|
| Accession number, ANO: | | | |
| Identification number, INO: | | | |
| Document type, DT: | Monographic publication | | |
| Type of record, TR: | Textual printed material | | |
| Contents code, CC: | Bachelor Thesis | | |
| Author, AU: | Nemanja Lazukić | | |
| Mentor, MN: | Ilija Bašičević, PhD | | |
| Title, TI: | Implementation of users and content management system in Digital television | | |
| Language of text, LT: | Serbian | | |
| Language of abstract, LA: | Serbian | | |
| Country of publication, CP: | Republic of Serbia | | |
| Locality of publication, LP: | Vojvodina | | |
| Publication year, PY: | 2021. | | |
| Publisher, PB: | Author's reprint | | |
| Publication place, PP: | Novi Sad, Dositeja Obradovica sq. 6 | | |
| Physical description, PD: (chapters/pages/ref./tables/pictures/graphs/appendices) | 7/49/6/23/26/0/0 | | |
| Scientific field, SF: | Electrical Engineering | | |
| Scientific discipline, SD: | Computer Engineering, Engineering of Computer Based Systems | | |
| Subject/Key words, S/KW: | Digital television, Rest-API, HTTP protocol, Data base | | |
| UC | | | |
| Holding data, HD: | The Library of Faculty of Technical Sciences, Novi Sad, Serbia | | |
| Note, N: | | | |
| Abstract, AB: | This paper describes one solution for user, device and content management system. It is made out of two main parts front-end and back-end. Front-end is responsible for sending requests towards back-end, accepting responses and representation of incoming data. Back-end is responsible for processing incoming requests, reading and writing to the database and sending the response to front-end. | | |
| Accepted by the Scientific Board on, ASB: | | | |
| Defended on, DE: | | | |
| Defended Board, DB: | President: | Miroslav Popović, PhD | |
| | Member: | Nebojša Pjevalica, PhD | Menthor's sign |
| | Member, Mentor: | Ilija Bašičević, PhD | |

Zahvalnost

Zahvaljujem se mentoru, prof. dr Iliji Bašičeviću i tehničkom mentoru, Radenku Banoviću, na stručnoj pomoći, savjetima i strpljenju tokom izrade ovog rada.

Takođe, zahvaljujem se RT-RK institutu, na prilici za realizaciju diplomskog rada.

Na kraju, posebna zahvalnost porodici i prijateljima, za svu podršku koju su mi neizmjerno pružali tokom procesa školovanja.



УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



SADRŽAJ

| | | |
|-------|--|----|
| 1. | Uvod | 1 |
| 2. | Teorijske osnove..... | 1 |
| 2.1 | REST API | 1 |
| 2.2 | HTTP protokol..... | 3 |
| 2.3 | JSON | 4 |
| 2.4 | JWT..... | 5 |
| 2.5 | <i>Python</i> | 6 |
| 2.6 | <i>SQLite</i> | 7 |
| 2.7 | <i>VueJS</i> | 7 |
| 3. | Koncept rješenja | 9 |
| 3.1 | <i>Back-end</i> | 9 |
| 3.1.1 | Arhitektura baze podataka..... | 10 |
| 3.1.2 | Serverska aplikacija..... | 10 |
| 3.2 | <i>Front-end</i> | 12 |
| 3.3 | Primjer komunikacije sistema..... | 12 |
| 4. | Programsko rješenje | 14 |
| 4.1 | Realizacija baze podataka | 14 |
| 4.2 | Realizacija serverske aplikacije | 15 |
| 4.2.1 | Entiteti | 15 |
| 4.2.2 | Sloj za pristup podacima | 17 |
| 4.2.3 | Kontroleri | 21 |
| 4.3 | Realizacija klijentske aplikacije..... | 26 |
| 5. | Ispitivanje realizovanog sistema..... | 41 |

| | | |
|-----|---|----|
| 5.1 | Ispitivanje funkcionalnosti upravljanja korisnicima | 41 |
| 5.2 | Ispitivanje funkcionalnosti upravljanja uređajima..... | 44 |
| 5.3 | Ispitivanje funkcionalnosti upravljanja paketima i kanalima | 45 |
| 5.4 | Ispitivanje funkcionalnosti upravljanja administratorima | 46 |
| 6. | Zaključak | 48 |
| 7. | Literatura | 49 |

SPISAK SLIKA

| | |
|---|----|
| Slika 2.1 Rest API..... | 2 |
| Slika 2.2 HTTP odgovor (lijevo) i HTTP zahtjev (desno) | 4 |
| Slika 2.3 Formiranje JSON objekt tipa..... | 5 |
| Slika 2.4 JSON objekt..... | 5 |
| Slika 2.5 JWT token..... | 6 |
| Slika 3.1 Arhitektura rješenja | 9 |
| Slika 3.2 Sistem rada JWT tokena | 11 |
| Slika 3.3 Jedan primjer komunikacije unutar sistema | 13 |
| Slika 4.1 Model baze podataka | 14 |
| Slika 4.2 Home stranica | 27 |
| Slika 4.3 Registration stranica | 28 |
| Slika 4.4 Login stranica | 29 |
| Slika 4.5 Menu stranica | 30 |
| Slika 4.6 User stranica | 32 |
| Slika 4.7 AddDeleteUser stranica | 33 |
| Slika 4.8 ChangeUser stranica | 34 |
| Slika 4.9 Devices stranica..... | 35 |
| Slika 4.10 PackageChannels stranica..... | 37 |
| Slika 4.11 Admin stranica..... | 38 |
| Slika 4.12 AddDeleteAdmin stranica | 39 |
| Slika 4.13 ChangeAdmin stranica | 40 |
| Slika 5.1 Upravljanje korisnicima - Izlistavanje korisnika..... | 43 |
| Slika 5.2 Upravljanje korisnicima - Izlistavanje korisničkih uređaja i pretplata..... | 44 |

| | |
|---|----|
| Slika 5.3 Izlistavanje uređaja | 45 |
| Slika 5.4 Izlistavanje kanala i paketa u sistemu..... | 46 |
| Slika 5.5 Izlistavanje administratora..... | 47 |

SPISAK TABELA

| | |
|--|----|
| Tabela 4.1 Pristup podacima o korisniku..... | 18 |
| Tabela 4.2 Pristup podacima o uređaju..... | 19 |
| Tabela 4.3 Pristup podacima o kanalu | 19 |
| Tabela 4.4 Pristup podacima o paketu | 20 |
| Tabela 4.5 Pristup podacima o administratoru | 21 |
| Tabela 4.6 User controller | 23 |
| Tabela 4.7 Device controller..... | 23 |
| Tabela 4.8 Channel controller..... | 24 |
| Tabela 4.9 Package controller..... | 25 |
| Tabela 4.10 Administrator controller..... | 26 |
| Tabela 4.11 Funkcije Menu stranice..... | 29 |
| Tabela 4.12 Funkcije User stranice..... | 31 |
| Tabela 4.13 Funkcije AddDeleteUser stranice | 32 |
| Tabela 4.14 Funkcije ChangeUser stranice | 34 |
| Tabela 4.15 Funkcije Devices stranice | 35 |
| Tabela 4.16 Funkcije PackageChannels stranice | 36 |
| Tabela 4.17 Funkcije Admin stranice | 38 |
| Tabela 4.18 Funkcije AddDeleteAdmin stranice..... | 38 |
| Tabela 4.19 Funkcije ChangeAdmin stranice..... | 40 |
| Tabela 5.1 Ispitivanje funkcionalnosti upravljanja korisnicima..... | 42 |
| Tabela 5.2 Ispitivanje funkcionalnosti upravljanja uređajima..... | 44 |
| Tabela 5.3 Ispitivanje funkcionalnosti upravljanja paketima i kanalima | 46 |
| Tabela 5.4 Ispitivanje funkcionalnosti upravljanja administratorima | 47 |

SKRAĆENICE

DTV - *Digital Television* - Digitalna televizija.

REST - *Representational state transfer* - Prenos prikaza stanja.

API - *Application programming interface* - Programski interfejs aplikacije.

HTTP - *Hypertext Transfer Protocol* - Protokol za prenos hiperteksta.

XML - *Extensible Markup Language* - Proširivi jezik za označavanje.

HTML - *Hypertext Markup Language* - Jezik za označavanje hiperteksta.

JSON - *JavaScript Object Notation* - Tekstualni format za razmenu podataka.

CRUD - acronym for: *Create, Read, Update, Delete* - Akronim za: kreiranje, čitanje, ažuriranje, brisanje.

OSI - *Open System Interconnection* - Otvoreno povezivanje sistema.

TCP - *Transmision Control Protocol* - Transmisioni kontrolni protokol.

IP - *Internet Protocol* - Internet protokol.

MIME - *Multipurpose Internet Mail Extensions* - Višenamjenska internet mejl ekstenzija.

JWT - *JSON Web Token* - JSON žeton.

DOM - *Document Object Model* - DOM stablo.

SFC - *Single-File Components*.

AJAX - *Asynchronous JavaScript and XML* - Asinhroni JavaScript i XML.

IDE - *Integrated development environment* - Integrisano razvojno okruženje.

1. Uvod

Prva analogna televizijska tehnologija javlja se 1920. godine, u početku tek kao eksperimentalni koncept bez komercijalne upotrebe. Sredinom dvadesetog vijeka dolazi do pojave prvih crno-bijelih monohromatskih televiziora dostupnih široj javnosti. Prije pojave televizije, glavna sredstva informisanja stanovništva bili su radio i štampani mediji, ali televizija kao revolucionarno otkriće koje spaja sliku i zvuk i prenosi ih do korisnika brzo uzima primat i postaje najvažniji masovni medij, kao i glavno sredstvo informisanja i zabave javnosti. Nova tehnologija doživljava mnoge promjene i poboljšanja u narednim decenijama, a jedna od prekretica je pojava televizora u boji koji su doživljaj gledanja televizije još više približili gledaocima [1].

Napretkom računarske moći i pristupačnosti računara dolazi do pojave digitalne televizije DTV (*Digital Television*) koja predstavlja jedan od najznačajniji pomaka televizijske tehnologije. DTV odlikuje bolji kvalitet zvuka, visoka rezolucija slike, interaktivnost, otpornost digitalnog signala na smetnje, manje zauzeće prenosnog kanala. Od 2000. godine počinje prelazak na digitalni televizijski signal širom svijeta, te je do 2021. proces digitalizacije u čitavom svijetu skoro potpuno završen [2].

Rasprotranjenost i pristupačnost DTV-a dovodi do sve većeg broja korisnika, ali i sve većeg obima sadržaja koji se nudi, što uslovljava potrebu za sveobuhvatnim sistemom koji će kontrolisati i upravljati kako korisnicima tako i sadržajem.

U ovom radu opisan je sistem za upravljanje korisnicima, uređajima i sadržajem, sastavljen iz dvije cjeline. *Back-end*-a, koji čine serverska aplikacija implementirana u *Python* programskom jeziku, *SQLite* baza podataka i *front-end*-a, web aplikacije, koja je implementirana u *VueJS* radnom okviru. Korisnik kroz web aplikaciju šalje zahtjeve za upravljanje korisnicima, uređajima i sadržajem u sistemu. *Back-end* je zadužen za primanje

zahtjeva, obradu zahtjeva, upis ili čitanje iz baze podataka i vraćanje odgovarajućeg odgovora web aplikaciji, koja zatim korisniku prikazuje odgovarajuće podatke ili poruku.

Rad je organizovan u sljedeće cjeline:

- Teorijske osnove – pregled tehnologija korišćenih u izradi zadatka.
- Koncept rješenja – idejni opis rješenja.
- Programsко rješenje – detaljan opis implementacije rješenja.
- Rezultati – prikaz rezultata testiranja.
- Zaključak – opis ispunjenosti zadatka i prijedlozi budućeg unapređenja.

2. Teorijske osnove

U ovom poglavlju izložene su teorijske osnove na kojima je zasnovan rad. Prvo je opisan *back-end*, pojam *REST (Representational state transfer)* API-ja (*Application programming interface*), osnove HTTP (*Hypertext Transfer Protocol*) protokola, JSON (*JavaScript Object Notation*) tip podataka, JWT (*JSON Web Token*) tokeni, *Python* i *SQLite*. Zatim je opisan *front-end*, *VueJS* radni okvir.

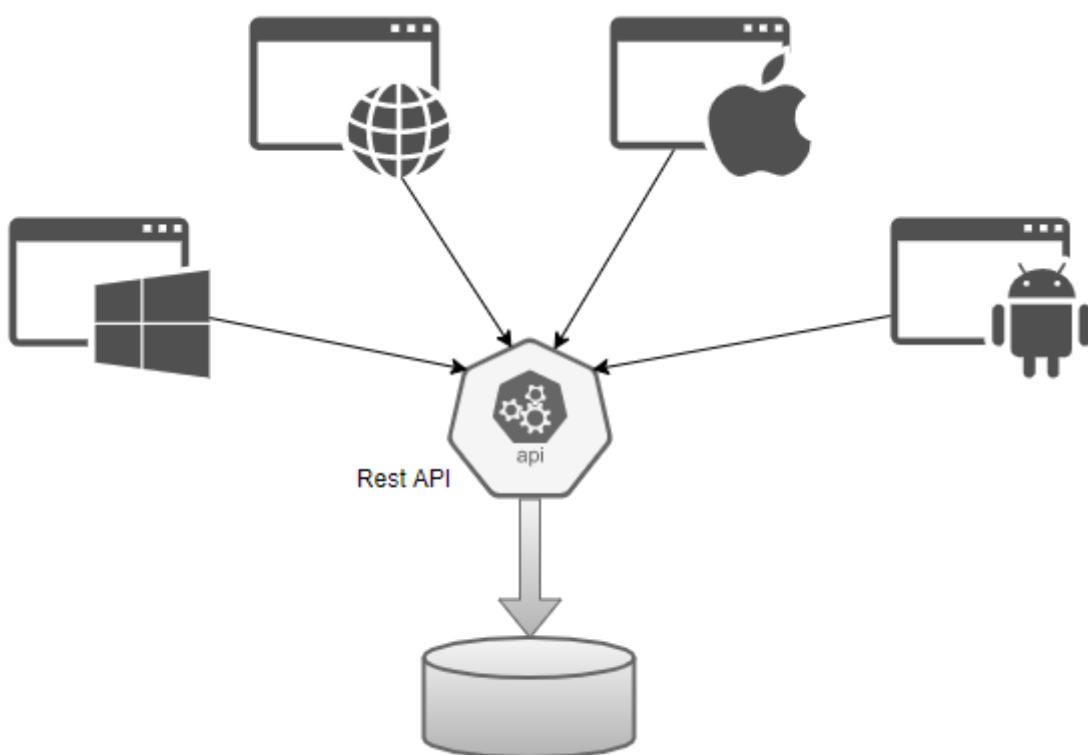
2.1 REST API

Programski interfejs aplikacije (eng. API) predstavlja softver koji omogućava komunikaciju i razmjenu resursa između aplikacija. Pri kreiranju API-ja definišu se načini komunikacije, protokol koji se koristi, format zahtjeva i odgovora, tip podataka koji će se koristiti itd. Ovakvi interfejsi se prave tako da izvršavaju jedan skup operacija, ali u isto vrijeme se ostavlja mogućnost promjene i proširenja u svrhu izvršavanja specifičnih zahtjeva programera. API omogućava pristup objektima ili funkcionalnostima određenog sistema bez potrebe za poznavanjem konkretnе implementacije u samom sistemu. Neki od primjera ovog interfejsa su API za softverske biblioteke, API između aplikacija i operativnog sistema, *web API* [3].

Prenos prikaza stanja (eng. REST) predstavlja stil arhitekture aplikacija, koji za cilj ima što lakšu komunikaciju između računarskih sistema na internetu. Implementacija klijenta i servera je potpuno nezavisna, što znači da izmjene koda na klijentskoj, odnosno serverskoj strani neće direktno uticati na promjenu ponašanja klijenta, odnosno servera. Sve komponente u sistemu implementiraju određeni format zahtjeva i odgovora što omogućava očuvanje modularnosti i nezavisnu evoluciju komponenti. Više klijenata može da pristupa istom

serveru sa različitih sistema. Jedna od glavnih karakteristika ovakvih sistema je nedostatak stanja, što implicira da komponente ne moraju da znaju međusobno stanje da bi izvršile operacije [4].

Rest API predstavlja *web API* koji implementira REST stil arhitekture. Sistem se sastoji od klijenta, servera i resursa. Klijent šalje zahtjeve da bi dobio ili izmijenio određene resurse, na što server vraća odgovor. Sam stil arhitekture ne specificira koji protokol se koristi za komunikaciju, ali je to najčešće HTTP protokol. Format poruka može biti XML (*Extensible Markup Language*), HTML (*Hypertext Markup Language*), JSON itd. Sistem je bez stanja, zahtjevi su međusobno nezavisni i svaki pojedinačni zahtjev sadrži sve potrebne informacije za izvršenje na serveru. Takođe, sistem se pravi sa slojevitom arhitekturom, svaki podsistem komunicira sa ostalim podsistemima, ali u isto vrijeme njegova funkcionalnost je odvojena od ostalih. Uniformnost je jedna od najvažnijih karakteristika, jer omogućava različitim klijentima (*web* aplikacija, android aplikacija, desktop aplikacija, iPhone aplikacija itd.) da koriste API, bez posebnog prilagođenja samog API-ja, što je prikazano na slici 2.1. Iz tog razloga koriste se standardizovani načini komunikacije npr. zahtjevi se šalju putem HTTP protokola, koriste se CRUD (*Create, Read, Update, Delete*) operacije i JSON format za prenos podataka.



Slika 2.1 Rest API

2.2 HTTP protokol

HTTP (eng. *Hypertext Transfer Protocol*) je najrasprostranjeniji internet protokol koji se danas koristi i kao takav je korišćen tokom izrade ovog rada. Protokol pripada aplikativnom sloju OSI (*Open System Interconnection*) referentnog modela, a njegova osnovna namjena je isporučivanje HTML dokumenata. Komunikacija se obavlja po sistemu klijent – server, gdje klijent šalje zahtjeve, a server vraća odgovore. Server osluškuje na određenom portu, a klijent inicira prenos podataka uspostavljanjem TCP (*Transmision Control Protocol*)/IP (*Internet Protocol*) veze sa serverom. Klijent šalje zahtjev koji se sastoji od:

- zaglavlja
 - HTTP metod
 - naziv traženog resursa
 - verzija HTTP-a
 - opcionih dodatnih polja zaglavlja (*Content-Length*, *Content-Type*, *Accepted-Language*, itd.)
- prazne linije
- tijela zahtjeva, opciono

U HTTP protokolu definisan je niz metoda koje označavaju koju operaciju klijent želi da izvrši sa resursima. Neke od metoda su: GET – dobavljanje određenog resursa, POST – traži od servera da prihvati i najčešće sačuva podatke poslane u tijelu zahtjeva, DELETE – brisanje određenog resursa, PATCH – modifikacije postojećeg resursa itd. Primjer jednog HTTP zahtjeva prikazan je na slici 2.2.

Ukoliko je zahtjev u ispravnoj formi dolazi do njegove obrade na serveru i vraćanja odgovora klijentu. Odgovor se sastoji od:

- zaglavlja
 - verzija HTTP-a
 - statusni kod
 - opcionih dodatnih polja zaglavlja
- prazne linije
- tijela odgovora, opciono

Statusni kod HTTP odgovora predstavlja trocifreni kod i kratki opisni tekst. Postoji pet podgrupa statusnih kodova: 1xx – informativni kodovi, 2xx – tražena akcija je uspješna, 3xx – preusmjerenje, 4xx – greška na klijentskoj strani, 5xx – greška na serverskoj strani. Konkretni primjeri statusnih kodova: 100 *Continue*, 200 *OK*, 301 *Moved Permanently*, 400 *Bad Request*, 404 *Not Found*, 500 *Internal Server Error* [5]. Primjer jednog HTTP odgovora prikazan je na slici 2.2.

The image shows two terminal windows side-by-side. The left window displays an HTTP response (200 OK) with headers and a JSON body. The right window displays an HTTP request (POST /login) with various headers and a URL parameter.

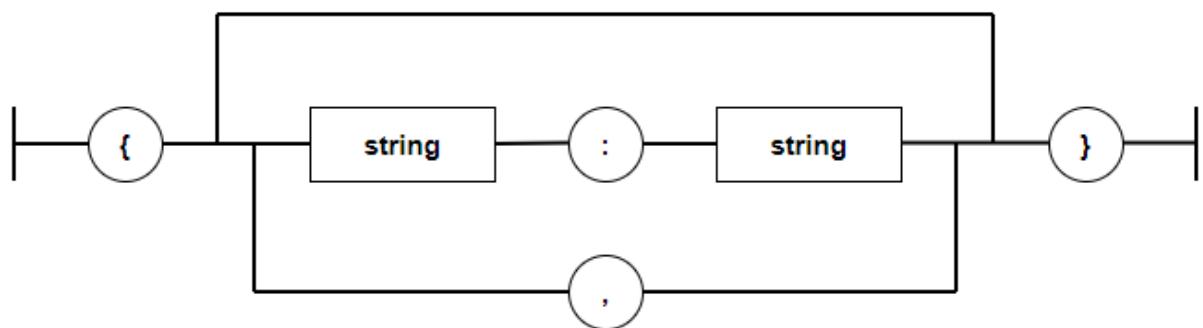
| |
|---|
| HTTP/1.1 200 OK |
| Date: Sun, 10. April 2020, 10:43:12 GMT |
| Server: Apache/2.0.4 (Win32) |
| Content-Length: 24 |
| Connection: Closed |
| {"response": "Successful"} |

| |
|-----------------------------|
| POST /login HTTP/1.1 |
| Host: www.example.com |
| Accept-Language: en |
| Content-Length: 30 |
| User-Agent: Mozilla/4.0 |
| user=ex&pass=ample |

Slika 2.2 HTTP odgovor (lijevo) i HTTP zahtjev (desno)

2.3 JSON

JSON predstavlja tekstualni format podataka koji se zbog jednostavnosti i razumljivosti najčešće koristi za razmjenu podataka na internetu. Prije njegove pojave za razmjenu podataka najčešće je korišten XML tip, od koga se polako odustaje, jer zahtjeva tagove koji čine čitav zapis dužim i teže čitljivim. Inspirisan je *JavaScript* programskim jezikom, ali je jezički nezavisno i većina programskih jezika posjeduje ugrađene funkcije ili funkcije dostupne preko biblioteka za parsiranje i generisanje JSON-a. Ekstenzija ovog tipa podataka je *.json*, a MIME (*Multipurpose Internet Mail Extensions*) format je *application/json*. Podržava više tipova podataka: broj, niz karaktera, bulove vrijednosti, niz, objekat i nula vrijednost. Opšti oblik JSON-a je { “ime”: “vrijednost”} [6]. Primjer formiranja JSON objekt tipa je prikazan na slici 2.3, a primjer jednog JSON objekta je prikazan na slici 2.4.



Slika 2.3 Formiranje JSON objekt tipa



Slika 2.4 JSON objekt

2.4 JWT

Prilikom pristupa internet servisima najčešće je potrebna neka forma prijavljivanja, sa korisničkim imenom ili imejlom i lozinkom, da bi se pristupilo određenim resursima. Da klijent ne bi morao sa svakim zahtjevom slati svoje kredencijale, koriste se JWT tokeni. JWT token predstavlja internet standard koji omogućava sigurnu komunikaciju i razmjenu resursa. Prilikom inicijalnog prijavljivanja klijenta, server formira JWT token i prosljeđuje ga klijentu, koji će token slati u svakom narednom zahtjevu i tako omogućiti serveru da identificuje klijenta bez potrebe za ponovnim prijavljivanjem.

Da bi se obezbijedila sigurnost resursa, tokeni su digitalno potpisani, koristeći tajni ključ ili parom javni/tajni ključ.

Ovaj vid zaštite je u širokoj upotrebi jer tokeni zauzimaju malo memorije i efikasni su za transport. Mogu se prenijeti kao dio URL-a, kao jedan od parametara u POST metodi ili u HTTP zaglavljima [7].

JWT token se sastoji od:

- zaglavlja (eng. *Header*) – opisuje algoritam korišćen za formiranje tokena i tip tokena
- polja za podatke (eng. *Payload*)
- potpis (eng. *Signature*) – verifikacija

Primjer jednog JWT tokena je prikazan na slici 2.5.



Slika 2.5 JWT token

2.5 Python

Python je dinamički tipiziran programski jezik, visokog nivoa i opšte namjene. Inspirisan je idejom o jednostavnom programiranju, njegova sintaksa je razumljiva i jednostavna za učenje. Umjesto vitičastih zagrada, izdvajanje blokova koda se vrši uvlačenjem, umjesto znakova interpunkcije koriste se riječi iz engleskog jezika, na primjer možemo koristi ključne riječi *is* i *is not* umjesto “==” i “!=”.

Podržava imperativni, objetno-orientisani, funkcionalni, strukturalni i reflektivni stil programiranja. U osnovi on je interpretativni programski jezik, ali postoje i pravi prevodioci koji ga prevode u mašinski jezik. *Python* posjeduje veliki broj standardnih biblioteka širokog spektra namjene: podrška za rad sa standardnim internet formatima i protokolima, moduli za kreiranje grafičkih korisničkih interfejsa, povezivanje sa relacionom bazom podataka, generisanje slučajnih brojeva, itd.

Trenutno *Python* predstavlja jedan od najzastupljenijih programskih jezika u svijetu, a neke od velikih organizacija koje ga koriste su: *Google, NASA, CERN, Amazon* [8].

2.6 *SQLite*

SQLite je biblioteka otvorenog koda implementirana u *C* programskom jeziku i predstavlja relacioni sistem za upravljanje bazama podataka. Koristi *PostgreSQL* sintaksu, ali za razliku od ostalih SQL baza podataka ne koristi klijent server arhitekturu, nego je ugrađena u samu aplikaciju i čita i piše direktno na disk. Funkcionalnosti baze podataka se koriste putem jednostavnih poziva funkcija, što ovo rješenje čini bržim od ostalih. Za razliku od klasičnih sistema sa klijent server arhitekturom, *SQLite* zahtjeva mnogo manje konfiguracija. Kontrola pristupa se vrši putem dozvola fajl sistema, datih bazi podataka, bez potrebe za dodatnim procesima koji provjeravaju prava pristupa. Takva implementacija omogućava veoma brzo čitanje iz baze podataka što je čini idealnim rješenjem za sisteme koji zahtijevaju jednostavne upite. Sa druge strane, zbog nedostatka posebnih procesa za provjeru prava pristupa, prilikom pisanja u bazu dolazi do zaključavanja, što onemogućava više različitih procesa da upisuju u bazu podataka u isto vrijeme.

Distributeri *SQLite* sistema pružaju samostalne programe koji služe za: kreiranje baze podataka, definisanje tabele, dodavanje i mijenjanje redova, pokretanje upita i upravljanje fajlom baze podataka.

Trenutno *SQLite* je najrasprostranjenija baza podataka u svijetu. Koristi se u raznim pretraživačima: *Opera*, *Google Chrome*, *Mozilla*, veb radnim okvirima: *Django*, *Ruby on Rails*, *Laravel* i mnogim drugim softverima: *Skype*, *BMW IDrive*, *Symbian OS*, *Windows 10* itd. [9].

2.7 *VueJS*

VueJS je *front-end JavaScript* radni okvir za kreiranje korisničkog interfejsa i *single-page* aplikacija, otvorenog koda. Glavna karakteristika ovog radnog okvira je prilagodljiva arhitektura, koja se fokusira na deklarativno renderovanje i prilagodljive komponente. Službena stranica *VueJS*-a pruža biblioteke i pakete za rad sa kompleksnijim operacijama kao što su: rutiranje (eng. *routing*), upravljanje stanjima (eng. *state management*) i alati za bildovanje (eng. *build tools*). Osnovna gradivna jedinica *Vue* radnog okvira su komponente, koje predstavljaju specifične elemente kojima kompjajler dodaje određeno ponašanje.

Vue šabloni su bazirani na HTML-u i mogu se parsirati u pretraživača ili HTML parseru. Jedna od glavnih prednosti *Vue* radnog okvira je virtuelni DOM (*Document Object Model*), koji omogućava renderovanje komponenti unutar sopstvene memorije, prije izmjena

u samom pretraživaču. Pomoću toga i reaktivnog sistema, *Vue* identificuje minimalni broj komponenti koje treba ponovo renderovati što poboljšava iskorištenost memorije i povećava brzinu rada aplikacije.

Neki od popularnih alata koje pruža *Vue* su:

- *Devtools* – dodaci za pretraživače, koji služe za otklanjanje grešaka *Vue* aplikacija.
- *Vue CLI* – softver za razvoj aplikacija.
- *Vue Loader* – omogućava pravljanje *Vue* komponenti u SFC (*Single-File Components*) formatu.

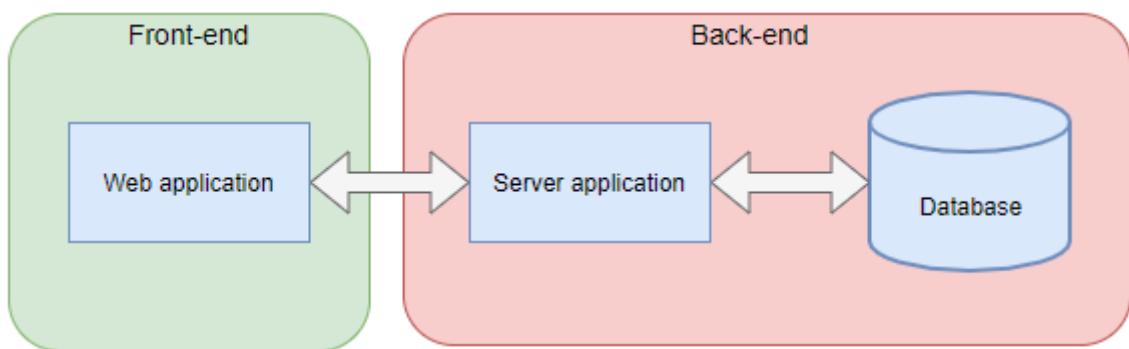
Neki od popularnih biblioteka koje pruža *Vue* su:

- *Vue Router* – službena biblioteka za rutiranje.
- *Vuex* – menadžer stanja *Vue* aplikacije.
- *Vue Server Renderer* – renderovanje na serverskoj strani aplikacije.

3. Koncept rješenja

U ovom poglavlju dat je opis koncepta rješenja sistema za upravljanje korisnicima i sadržajem u digitalnoj televiziji. Ova aplikacija je namijenjena provajderima DTV usluga, sa ciljem kontrole, praćenja i manipulacije korisnicima, uređajima i sadržajem koji se pruža.

Rješenje predstavlja *full-stack* aplikaciju koja se sastoji od dvije šire cjeline *back-end-a* i *front-end-a*. Prikazano na slici 3.1. U nastavku je opisana arhitektura ovih cjelina, njihova funkcionalnost i međusobna komunikacija.



Slika 3.1 Arhitektura rješenja

3.1 *Back-end*

Back-end predstavlja serversku stranu sistema, zadužen za primanje zahtjeva od strane klijenta, obradu zahtjeva, vraćanje odgovarajućeg odgovora, skladištenje podataka i očuvanje sigurnosti čitavog sistema. Sastavljen je iz dvije cjeline, baze podataka i *Rest API-ja*.

3.1.1 Arhitektura baze podataka

Baza podataka predstavlja dio sistema zadužen za skladištenje i očuvanje podataka. Prilikom projektovanja same baze posebnu pažnju treba posvetiti tipu baze koja će se koristiti, tipovima i strukturi podataka koji će se skladištiti i šemi baze. U ovom rješenju implementirana je relaciona baza podataka, sastavljena od pet glavnih tabela i tri asocijativne tabele, korištene za realizaciju više prema više (eng. *many-to-many*) veze između tabela.

Prilikom implementacije sistema za rukovanje korisnicima i sadržajem postoji nekoliko informacija koje je potrebno skladištiti. Svakom sistemu su potrebni administratori, koji u ovoj implementaciji predstavljaju i krajne korisnike aplikacije. Podaci o administratorima se skladište u tabeli *administrator*. Takođe, potrebno je čuvati i podatke o samim korisnicima usluga provajdera, za šta se koristi tabela *user*. Podaci o uređajima koje provajder ima na raspolaganju i koji su dodijeljeni korisnicima se skadište u tabeli *device*. Sadržaj koji se pruža od strane provajdera je koncipiran kroz dvije tabele, *channel*, koja čuva podatke o dostupnim kanalima i *package*, koja čuva podatke o dostupnim paketima.

3.1.2 Serverska aplikacija

Serverska aplikacija je implementirana u tri cjeline. Entiteti, predstavljaju klase koje se mapiraju u tabele baze podataka, sloj za pristup podacima, koji vrši dodavanje, brisanje i izvlačenje podataka iz baze i kontroleri, koji su zaduženi za prihvatanje zahtjeva, obradu zahtjeva, poziv funkcija iz sloja za pristup podacima, generisanje i slanje odgovora.

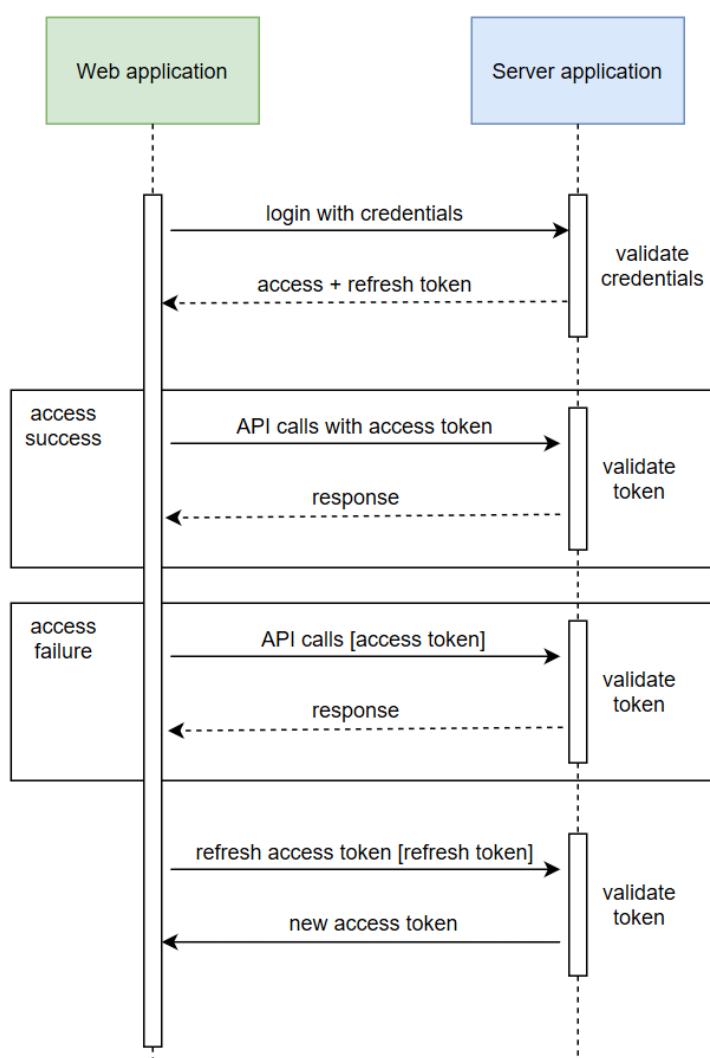
Aplikacija je napisana u *Python* programskom jeziku. *Rest* tehnologija je implementirana upotrebom *Flask* radnog okvira, takođe napisanog u *Python* programskom jeziku. *Flask* predstavlja web radni okvir, koji olakšava kreiranje web server aplikacija i posjeduje veliki ekosistem biblioteka. Komunikacija se vrši upotrebom HTTP protokola i JSON-a kao formata podataka za razmjenu.

Kao sistem za upravljanje bazom podataka odabrana je *SQLite* biblioteka, otvorenog koda, koja se lako mijenja u slučaju budućih unapređenja i posjeduje kompatibilnost unazad. Jednostavna je za korištenje i brza, jer se baza podataka čuva na samom disku u obliku fajla.

Posrednik u komunikaciji između *Python* programa i baze podataka je biblioteka *SQLAlchemy*, otvorenog koda, napisana u *Python* programskom jeziku. Biblioteka omogućava pojednostavljeni kreiranje baze podataka, mapiranjem *Python* objekata u tabele baze podataka. Takođe, obezbjeđuje jednostavan sistem kreiranja i izvršavanja upita.

Bezbjednost sistema je osigurana korištenjem JWT tokena, tačnije *Flask*-ovog dodatka *Flask-jwt-extended*, koji obezbjeđuje jednostavnije generisanje, kontrolu i razmjenu tokena. Prilikom prijavljivanja korisnika upotrebom *username*-a i *password*-a, server u odgovoru

prosljeđuje JWT pristupni token (eng. *access token*) i token za obnovu (eng. *refresh token*), koji se zatim skladišti u klijentskoj aplikaciji. Prilikom svakog zahtjeva ka serverskoj aplikaciji, klijent šalje i pristupni token u okviru HTTP zaglavla. Serverska aplikacija provjerava validnost tokena, nakon čega vraća odgovor. Pristupni token ima svoje vrijeme trajanja, kada dođe do isteka važenja tokena, klijentska aplikacija šalje zahtjev za obnovu pristupnog tokena, u vidu HTTP zahtjeva sa tokenom za obnovu u okviru HTTP zaglavla. Ukoliko je token za obnovu validan, serverska aplikacija vraća odgovor koji sadrži novi pristupni token. Dijagram sistema rada JWT tokena je prikazan na slici 3.2.



Slika 3.2 Sistem rada JWT tokena

3.2 **Front-end**

Front-end predstavlja klijentsku aplikaciju implementiranu uz pomoć VueJS radnog okvira, koncipiranu kroz dvanaest web stranica.

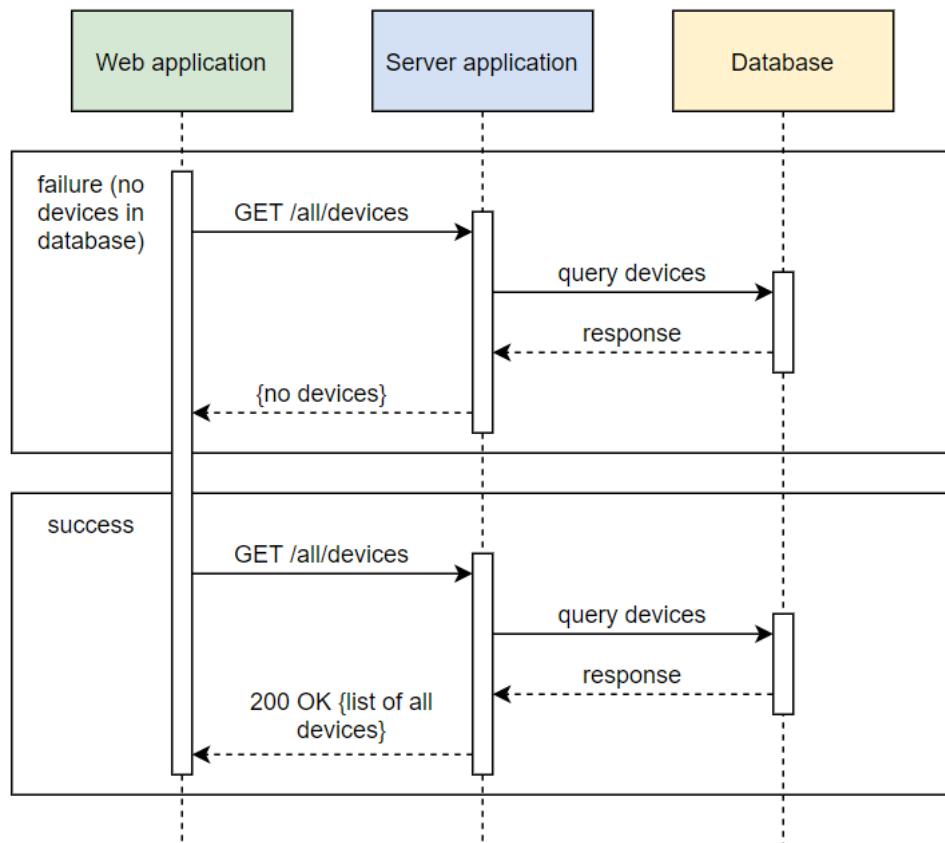
Slanje HTTP zahtjeva sa klijentske aplikacije i primanje HTTP odgovora od strane serverske aplikacije se vrši pomoću *Axios* biblioteke, koja je zasnovana na AJAX (Asynchronous JavaScript and XML) tehnologiji i koristi HTTP protokol.

Za skladištenje podataka kao sto su: JWT pristupni token i JWT token za obnovu, korištena je biblioteka *Vuex*, koja omogućava pristup stanju sistema iz bilo koje komponente uz unaprijed definisane načine dodavanja, brisanja i pristupa podacima. Skladištenje se vrši u formi ključ-vrijednost.

Rutiranje se vrši putem *Vue router* biblioteke, dostupne na oficijalnoj stranici *VueJS*-a. Za prikazivanje notifikacija korisniku korištena je *Vue-toastify* biblioteka, a za materijalni dizajn web aplikacije *Vuetify* radni okvir.

3.3 **Primjer komunikacije sistema**

Na slici: 3.3. prikazan je jedan primjer komunikacije unutar sistema. Web aplikacija šalje HTTP zahtjev, sa metodom GET, tražeći spisak svih dostupnih uređaja, serverska aplikacija provjerava da li u bazi podataka postoji traženi resurs, ako postoji, korisničkoj aplikaciji se u JSON formatu šalje spisak svih dostupnih uređaja.



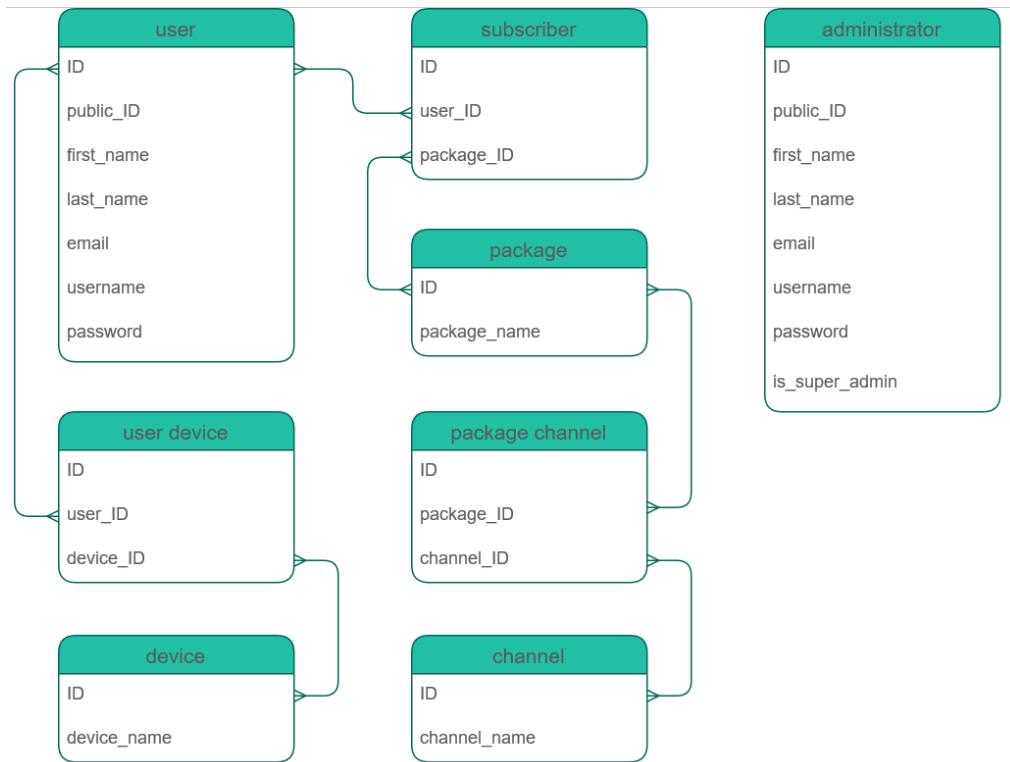
Slika 3.3 Jedan primjer komunikacije unutar sistema

4. Programsко rješenje

U ovom poglavlju opisan je način implementacije rješenja opisanog u prethodnom poglavlju. Prvo će biti opisana implementacija baze podaka, a zatim, serverske i klijentske aplikacije.

4.1 Realizacija baze podataka

Baza podataka je realizovana upotrebom *SQLite* biblioteke i *SQLAlchemy*, ORM mapera. Model baze podataka prikazan je na slici 4.1.



Slika 4.1 Model baze podataka

Baza podataka se sastoji od pet tabela:

- *User* tabela – skladišti sve potrebne podatke o korisniku (identifikator, javni identifikator, ime, prezime, imejl, nadimak, lozinka).
- *Device* tabela – skladišti podatke o uređaju (identifikator, ime uređaja).
- *Package* tabela – skladišti potrebne podatke o paketu (identifikator, ime paketa).
- *Channel* tabela – skladišti podatke o kanalu (identifikator, ime paketa).
- *Administrator* – skladišti sve potrebne podatke o administratoru (identifikator, javni identifikator, ime, prezime, imejl, nadimak, lozinku, informacija da li je administrator super admin).

Takođe postoje i tri asocijativne tabele:

- *User device* – skladišti identifikator korisnika i identifikator uređaja.
- *Package channel* – skladišti identifikator paketa i identifikator kanala.
- *Subscriber* – skladišti identifikator korisnika i identifikator paketa.

4.2 Realizacija serverske aplikacije

Serverska aplikacija je sastavljena iz tri cjeline: entiteti, koji predstavljaju klase koje se mapiraju u bazu podataka; sloja za pristup podacima, koji je zadužen za upis i čitanje iz baze podataka i kontroleri u kojima je implementirana logika same aplikacije. Takođe, sadrži i funkcije za definisanje baze podataka i kreiranje servera.

4.2.1 Entiteti

Entiteti predstavljaju klase koje definišu nazive tabele u bazi podataka, tipove i naziv podataka koji će predstavljati polja unutar tabela baze. Svaki zasebni entitet posjeduje `__repr__` funkciju koja definiše reprezentaciju objekta klase.

EntityUser:

- `id – integer`, identifikator, primarni ključ
- `public_id – integer`, javni identifikator
- `first_name – string`, ime korisnika
- `last_name – string`, prezime korisnika
- `email – string`, imejl adresa korisnika
- `username – string`, nadimak korisnika

-
- *password – string*, lozinka korisnika

EntityChannel:

- *id – integer*, identifikator, primarni ključ
- *channel_name – string*, naziv kanal

EntityPackage:

- *id – integer*, identifikator, primarni ključ
- *package_name – string*, naziv paketa

EntityDevice:

- *id – integer*, identifikator, primarni ključ
- *device_name – string*, naziv uređaja

EntityAdministrator:

- *id – integer*, identifikator, primarni ključ
- *public_id – integer*, javni identifikator
- *first_name – string*, ime administrator
- *last_name – string*, prezime administrator
- *email – string*, imejl administrator
- *username – string*, nadimak administrator
- *password – string*, lozinka administrator
- *is_super_admin – boolean*, oznaka super administrator

EntitySubscriber:

- *id – integer*, identifikator, primarni ključ
- *user_id – integer*, identifikator korisnika
- *package_id – integer*, identifikator paketa

EntityPackageChannel:

- *id – integer*, identifikator, primarni ključ
- *package_id – integer*, identifikator paketa
- *channel_id – integer*, identifikator kanala

EntityUserDevice:

- *id – integer*, identifikator, primarni ključ
- *user_id – integer*, identifikator korisnika
- *device_id – integer*, identifikator uređaja

4.2.2 Sloj za pristup podacima

Sloj za pristup podacima (eng. *Data Access Layer*) se koristi kako bi se pristup bazi podataka odvojio od ostatka aplikacije, što olakšava buduća unapređenja i osigurava nezavisnost cjelina unutar aplikacije. Ovakav koncept omogućava upis i čitanje iz baze podataka, iz ostatka koda, preko definisanih funkcija za čitanje i pisanje, bez direktnog pristupa bazi. U nastavku, detaljno će biti opisana svaka funkcija, njen naziv, ulazne i izlazne vrijednosti i funkcionalnost.

User:

| Naziv funkcije | Opis |
|--|---|
| <code>get_user_by_username(username)</code> | Vraća objekat korisnika, zadatog parametrom <i>username</i> . |
| <code>get_user_by_email(email)</code> | Vraća objekat korisnika, zadatog parametrom <i>email</i> . |
| <code>get_all_users()</code> | Vraća listu objekata svih korisnika. |
| <code>change_user_email(user, new_email)</code> | Mjenja postojeću <i>email</i> adresu korisnika, zadatog parametrom <i>user</i> , u adresu zadatu parametrom <i>new_email</i> . |
| <code>change_user_username(user, new_username)</code> | Mjenja postojeći <i>username</i> korisnika, zadatog parametrom <i>user</i> , u <i>username</i> zadat parametrom <i>new_username</i> . |
| <code>change_user_password(user, new_password)</code> | Mjenja postojeći <i>password</i> korisnika, zadatog parametrom <i>user</i> , u <i>password</i> zadat parametrom <i>new_password</i> . |
| <code>insert_user(first_name, last_name, username, email, password)</code> | Upisuje novog korisnika, sa kredencijalima zadatih parametrima: <i>first_name</i> , <i>last_name</i> , <i>username</i> , <i>email</i> , <i>password</i> . |
| <code>delete_user(user)</code> | Briše korisnika, zadatog parametrom <i>user</i> . |

| | |
|---|---|
| <code>get_all_user_devices(user_id)</code> | Vraća listu objekata uređaja, koje posjeduje korisnik zadat parametrom <code>user_id</code> . |
| <code>get_user_device(user_id, device_id)</code> | Vraća objekat uređaja, zadatog parametrom <code>device_id</code> , koji pripada korisniku zadatog parametrom <code>user_id</code> . |
| <code>insert_user_device(user_device)</code> | Upisuje novi korisnički uređaj, zadat objektom <code>user_device</code> . |
| <code>get_all_user_subscriptions(user_id)</code> | Vraća listu objekata svih paketa, na koje je korisnik pretplaćen. |
| <code>get_user_subscription(user_id, package_id)</code> | Vraća objekat paketa, zadatog parametrom <code>package_id</code> , na koji je pretplaćen korisnik zadat parametrom <code>user_id</code> . |
| <code>insert_subscription(user, package)</code> | Upisuje novi paket, zadat parametrom <code>package</code> , na koji se pretplaćuje korisnik zadat parametrom <code>user</code> . |
| <code>delete_subscription(user_package)</code> | Briše pretplatu korisnika na paket, zadat parametrom <code>user_package</code> . |

Tabela 4.1 Pristup podacima o korisniku

Device:

| Naziv funkcije | Opis |
|--|--|
| <code>get_device(device_name)</code> | Vraća objekat uređaja, zadatog parametrom <code>device_name</code> . |
| <code>get_device_by_id(device_id)</code> | Vraća objekat uređaja, zadatog parametrom <code>device_id</code> . |
| <code>get_all_devices()</code> | Vraća listu objekata svih uređaja. |
| <code>insert_device(device_name)</code> | Upisuje novi uređaj sa imenom zadatim |

| | |
|------------------------------|--|
| | parametrom <i>device_name</i> . |
| <i>delete_device(device)</i> | Briše uređaj, zadat parametrom <i>device</i> . |

Tabela 4.2 Pristup podacima o uređaju

Channel:

| Naziv funkcije | Opis |
|--------------------------------------|--|
| <i>get_channel(channel_name)</i> | Vraća objekat kanala, zadatog parametrom <i>channel_name</i> . |
| <i>get_channel_by_id(channel_id)</i> | Vraća objekat kanala, zadatog parametrom <i>channel_id</i> . |
| <i>get_all_channels()</i> | Vraća listu objekata svih kanala. |
| <i>insert_channel(channel_name)</i> | Upisuje novi kanal, sa imenom zadatim parametrom <i>channel_name</i> . |
| <i>delete_channel(channel)</i> | Briše kanal, zadat parametrom <i>channel</i> . |

Tabela 4.3 Pristup podacima o kanalu

Packages:

| Naziv funkcije | Opis |
|--------------------------------------|--|
| <i>get_package(package_name)</i> | Vraća objekat paketa, zadatog parametrom <i>package_name</i> . |
| <i>get_package_by_id(package_id)</i> | Vraća objekat paketa, zadatog parametrom <i>package_id</i> . |
| <i>get_all_packages()</i> | Vraća listu objekata svih paketa. |
| <i>insert_package(package_name)</i> | Upisuje novi paket, sa imenom zadatim parametrom <i>package_name</i> . |

| | |
|--|---|
| <i>delete_package(package)</i> | Briše kanal, zadat parametrom <i>package</i> . |
| <i>get_all_package_channels(package_id)</i> | Vraća listu objekata svih kanala, koje sadrži paket, zadat parametrom <i>package_id</i> . |
| <i>get_package_channel(channel_id, package_id)</i> | Vraća objekat kanala zadatog parametrom <i>channel_id</i> , koji se nalazi u paketu, zadatog parametrom <i>package_id</i> . |
| <i>insert_package_channel(channel, package)</i> | Upisuje novi kanal, zadat parametrom <i>channel</i> , u paket zadat parametrom <i>package</i> . |
| <i>delete_package_channel(channel_package)</i> | Briše kanal iz paketa, zadat parametrom <i>channel_package</i> . |

Tabela 4.4 Pristup podacima o paketu

Administrator:

| Naziv funkcije | Opis |
|---|---|
| <i>get_admin_by_username(username)</i> | Vraća objekat administratora, zadatog parametrom <i>username</i> . |
| <i>get_admin_by_email(email)</i> | Vraća objekat administratora, zadatog parametrom <i>email</i> . |
| <i>get_all_admins()</i> | Vraća listu objekata svih administratora. |
| <i>change_admin_email(admin, new_email)</i> | Mjenja postojeći <i>email</i> adresu administratora, zadatog parametrom <i>admin</i> , u adresu zadatu parametrom <i>new_email</i> . |
| <i>change_admin_username(admin, new_username)</i> | Mjenja postojeći <i>username</i> administratora, zadatog parametrom <i>admin</i> , u <i>username</i> zadat parametrom <i>new_username</i> . |
| <i>change_admin_password(admin,</i> | Mjenja postojeći <i>password</i> administratora, |

| | |
|---|---|
| <i>new_password)</i> | zadatog parametrom <i>admin</i> , u <i>password</i> zadat parametrom <i>new_password</i> . |
| <i>insert_admin(first_name, last_name, username, email, password, is_super_admin)</i> | Upisuje novog administratora, sa kredencijalima, zatim parametrima: <i>first_name, last_name, username, email, password, is_super_admin</i> . |
| <i>delete_admin(admin)</i> | Briše administratora, zadatog parametrom <i>admin</i> . |

Tabela 4.5 Pristup podacima o administratoru

4.2.3 Kontroleri

Kontroler predstavlja centralnu, kontrolnu jedinicu sistema. Zadužen je za primanje zahtjeva od strane klijentske aplikacije, obradu zahtjeva, poziv funkcija iz sloja za pristup podacima, obradu podataka i vraćanje odgovora klijentskoj aplikaciji. Ovo rješenje posjeduje pet kontrolera: userController, deviceController, channelController, packageController i administratorController, čije će funkcionalnosti detaljno biti opisana u nastavku.

Funkcije primaju HTTP zahtjeve, sa JSON formatom podataka, koji se parsira metodom *get_json()*. Na početku svake funkcije, provjerava se ispravnost JWT tokena pomoću funkcije *get_jwt_identity()*, ukoliko je token ispravan nastavlja se izvršavanje funkcije, ukoliko nije vraća se HTTP odgovor, sa 401 *Unauthorized* statusnim kodom. Sve funkcije koje će biti opisane u nastavku, u slučaju nemogućnosti ispunjenja određene akcije, šalju HTTP odgovor, sa odgovarajućom informacijom o grešci; u slučaju uspješne akcije, vraća se HTTP odgovor sa statusnim kodom i podacima u JSON obliku, u slučaju potražnje određenih resursa.

UserController:

| Naziv funkcije | Opis |
|-----------------------|--|
| <i>add_new_user()</i> | Prima zahtjev i izvlači korisničke podatke, poziva funkciju za dodavanje novog korisnika. U slučaju uspješne akcije, vraća 200 statusni kod. |

| | |
|----------------------------------|--|
| <i>delete_user()</i> | Prima zahtjev, izvlači korisnički <i>username</i> , poziva funkciju za brisanje korisnika. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>all_users()</i> | Poziva funkciju za dobijanje liste svih korisnika, ukoliko je akcija uspješna, šalje odgovor sa listom svih korisnika. |
| <i>change_email()</i> | Prima zahtjev, izvlači korisnički <i>username</i> i <i>new_email</i> , poziva funkciju za promjenu <i>email-a</i> . U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>change_username()</i> | Prima zahtjev, izvlači korisnički <i>username</i> i <i>new_username</i> , poziva funkciju za promjenu <i>username-a</i> . U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>change_password()</i> | Prima zahtjev, izvlači korisnički <i>username</i> i <i>new_password</i> , poziva funkciju za promjenu <i>password-a</i> . U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>check_user_devices()</i> | Prima zahtjev, izvlači korisnički <i>username</i> i poziva funkciju za dobijanje liste svih uređaja koje korisnik posjeduje. U slučaju uspješne akcije, šalje odgovor, sa listom svih uređaja. |
| <i>add_device_to_user()</i> | Prima zahtjev, izvlači korisnički <i>username</i> i <i>device_name</i> , poziva funkciju za dodavanje uređaja korisniku. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>delete_device_from_user()</i> | Prima zahtjev, izvlači korisnički <i>username</i> i <i>device_name</i> , poziva funkciju za brisanje korisnikovog uređaja. U slučaju uspješne |

| | |
|----------------------------------|--|
| | akcije, šalje 200 statusni kod. |
| <i>subscribe()</i> | Prima zahtjev, izvlači korisnički <i>username</i> i <i>package_name</i> , poziva funkciju za pretplaćivanje korisnika na paket. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>unsubscribe()</i> | Prima zahtjev, izvlači korisnički <i>username</i> i <i>package_name</i> , poziva funkciju za brisanje pretplate korisnika na paket. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>check_user_subscription()</i> | Prima zahtjev, izvlači korisnički <i>username</i> i poziva funkciju za dobijanje liste svih paketa na koje je korisnik pretplaćen. U slučaju uspješnosti akcije, šalje listu svih paketa na koje je korisnik pretplaćen. |

Tabela 4.6 User controller

DeviceController:

| Naziv funkcije | Opis |
|------------------------|--|
| <i>add_device()</i> | Prima zahtjev, izvlači <i>device_name</i> i poziva funkciju za dodavanje novog uređaja. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>delete_device()</i> | Prima zahtjev, izvlači <i>device_name</i> i poziva funkciju za brisanje uređaja. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>all_devices()</i> | Prima zahtjev, poziva funkciju za dobijanje liste svih uređaja. U slučaju uspješne akcije, šalje listu svih uređaja. |

Tabela 4.7 Device controller

ChannelController:

| Naziv funkcije | Opis |
|-------------------------|--|
| <i>add_channel()</i> | Prima zahtjev, izvlači <i>channel_name</i> i poziva funkciju za dodavanje novog kanala. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>delete_channel()</i> | Prima zahtjev, izvlači <i>channel_name</i> i poziva funkciju za brisanje kanala. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>all_channels()</i> | Prima zahtjev, poziva funkciju za dobijanje liste svih kanala. U slučaju uspješne akcije, šalje listu svih kanala. |

Tabela 4.8 Channel controller

PackageController:

| Naziv funkcije | Opis |
|-------------------------|--|
| <i>add_package()</i> | Prima zahtjev, izvlači <i>package_name</i> i poziva funkciju za dodavanje novog paketa. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>delete_package()</i> | Prima zahtjev, izvlači <i>package_name</i> i poziva funkciju za brisanje paketa. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>all_packages()</i> | Prima zahtjev, poziva funkciju za dobijanje liste svih paketa. U slučaju uspješne akcije, šalje listu svih paketa. |

| | |
|--------------------------------------|--|
| <i>add_channel_to_package()</i> | Prima zahtjev, izvlači <i>channel_name</i> i <i>package_name</i> , poziva funkciju za dodavanje kanala u paket. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>delete_channel_from_package()</i> | Prima zahtjev, izvlači <i>channel_name</i> i <i>package_name</i> , poziva funkciju za brisanje kanala iz paketa. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>check_package_channels()</i> | Prima zahtjev, izvlači <i>package_name</i> i poziva funkciju za dobijanje liste svih kanala u paketu. U slučaju uspješnosti akcije, šalje listu svih kanala koji se nalaze u paketu. |

Tabela 4.9 Package controller

AdministratorController:

| Naziv funkcije | Opis |
|-----------------------------|--|
| <i>add_new_admin()</i> | Prima zahtjev i izvlači administratorske podatke, poziva funkciju za dodavanje novog administratora. U slučaju uspješne akcije, vraća 200 statusni kod. |
| <i>delete_admin()</i> | Prima zahtjev, izvlači administratorski <i>username</i> , poziva funkciju za brisanje administratora. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>all_admins()</i> | Poziva funkciju za dobijanje liste svih administratora, ukoliko je akcija uspješna, šalje odgovor sa listom svih administratora. |
| <i>change_admin_email()</i> | Prima zahtjev, izvlači administratorski <i>username</i> i <i>new_email</i> , poziva funkciju za |

| | |
|---------------------------------|--|
| | promjenu <i>email</i> -a. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>change_admin_username()</i> | Prima zahtjev, izvlači administratorski <i>username</i> i <i>new_username</i> , poziva funkciju za promjenu <i>username</i> -a. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>change_admin_password()</i> | Prima zahtjev, izvlači administratorski <i>username</i> i <i>new_password</i> , poziva funkciju za promjenu <i>password</i> -a. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>admin_registration()</i> | Prima zahtjev, izvlači administratorske podatke, poziva funkciju za dodavanje novog administratora. U slučaju uspješne akcije, šalje 200 statusni kod. |
| <i>admin_login()</i> | Prima zahtjev, izvlači <i>username</i> i <i>password</i> , provjerava ispravnost kredencijala. U slučaju tačnih kredencijala, šalje odgovor koji sadrži JWT pristupni token i token za obnovu. |
| <i>check_admin_privileges()</i> | Prima zahtjev, izvlači <i>username</i> , provjerava da li je trenutni administrator, super administrator. Ako jeste, šalje 200 statusni kod. |
| <i>refresh()</i> | Prima zahtjev, izvlači JWT token za obnovu, ako je token validan, šalje odgovor sa novim pristupnim tokenom. |

Tabela 4.10 Administrator controller

4.3 Realizacija klijentske aplikacije

Klijentska aplikacija je implementirana koristeći *VueJS* radni okvir, uz upotrebu *Vue* oficijalnih biblioteka za rutiranje, *Vue router* i skladištenje podataka, *Vuex*.

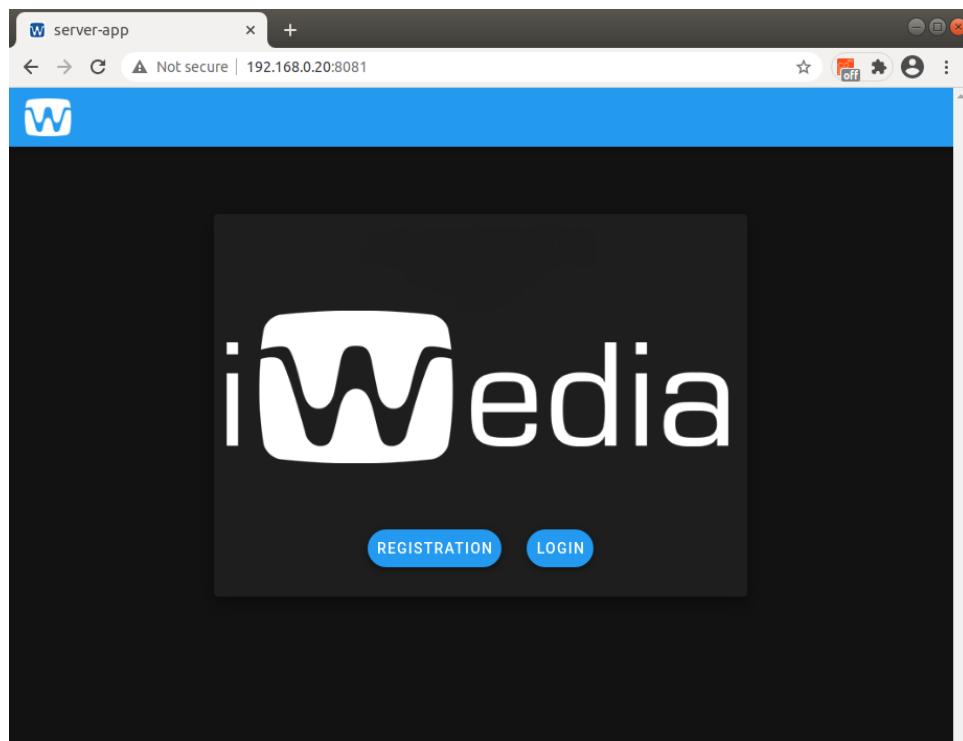
Aplikacija je realizovana kroz dvanaest web stranica: Home, Login, Registration, Menu, Users, AddDeleteUser, ChangeUser, Devices, PackageChannels, Admin, AddDeleteAdmin, ChangeAdmin. Stranice su napisane u HTML programskom jeziku, uz upotrebu *Vuetify* radnog okvira, a funkcionalnost aplikacije je realizovana u *JavaScript* programskom jeziku.

Zahtjevi ka serverskoj aplikaciji se šalju korištenjem *Axios* biblioteke, u zaglavlje HTTP zahtjeva se ubacuje pristupni JWT token.

Sve funkcije koje će biti opisane u nastavku šalju zahtjev na odgovarajuću rutu serverske aplikacije, zaduženu za opsluživanje konkretnе operacije. U slučaju nemogućnosti ispunjenja određene akcije, ili u slučaju negativnog odgovora serverske aplikacije, korisniku se ispisuje odgovarajuća poruka o grešci.

Home:

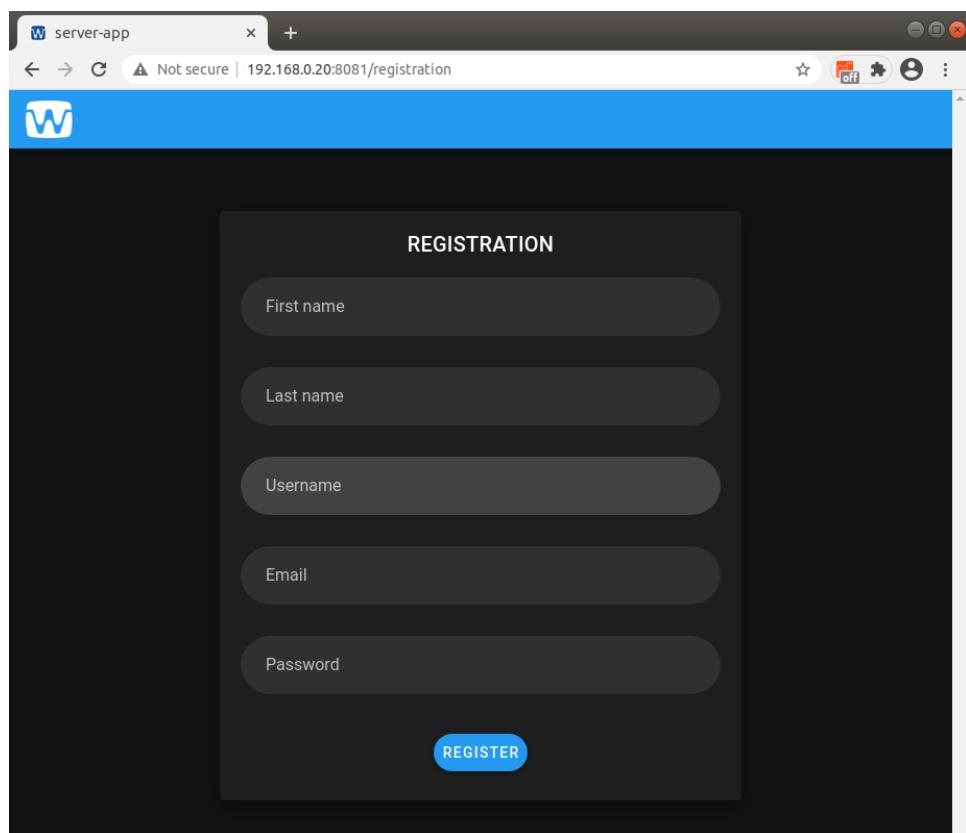
Stranica, predstavlja početnu stranicu aplikacije na kojoj se nalaze dva dugmeta, Registration i Login, pritiskom odgovarajućeg dugmeta pozivaju se funkcije *to_registration()*, koja korisnika vodi na Registration stranicu ili *to_login()*, koja korisnika vodi na Login stranicu. Na slici 4.2. prikazan je izgled Home stranice.



Slika 4.2 Home stranica

Registration:

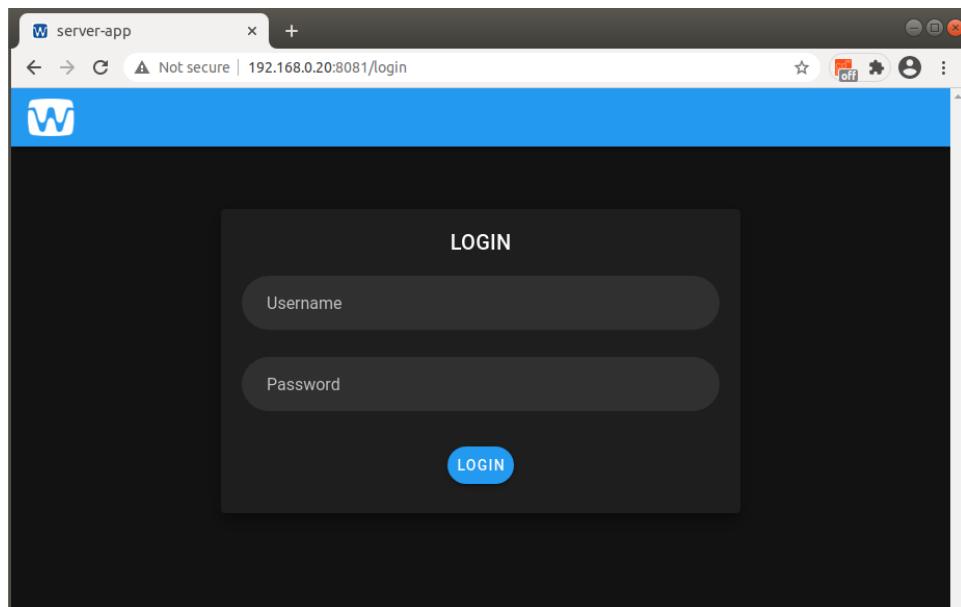
Stranica posjeduje polja za upis podataka administratora i sadrži jedno dugme, Register. Pritiskom na dugme aktivira se funkcija *registration()*, koja šalje zahtjev sa podacima o administratoru. U slučaju uspješne registracije, korisnik aplikacije se preusmjerava na Login stranicu. Na slici 4.3 je prikazan izgled Registration stranice.



Slika 4.3 Registration stranica

Login:

Stranica posjeduje polja za upis username-a i password-a i dugme Login. Pritiskom dugmeta aktivira se funkcija *login()*, koja šalje zahtjev sa kredencijalima administratora. U slučaju uspješne prijave, dobijeni JWT pristupni token i token za obnovu se skladište i korisnik se preusmjerava na Menu stranicu. Na slici 4.4 prikazan je izgled Login stranice.



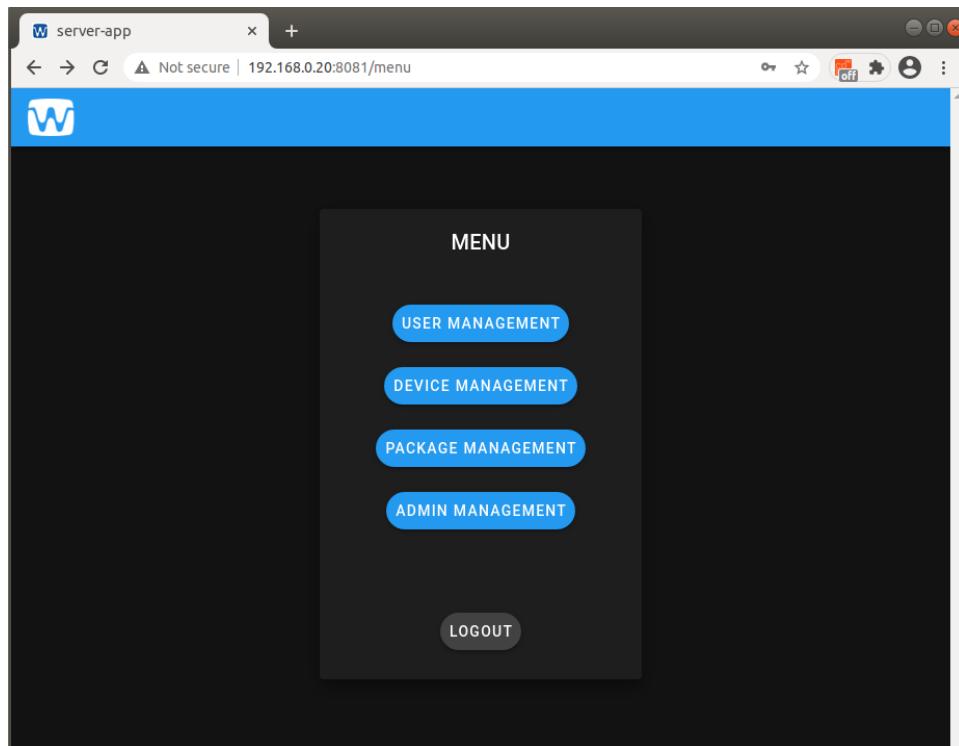
Slika 4.4 Login stranica

Menu:

Stranica posjeduje pet dugmadi: User Management, Device Management, Package Management, Admin Management i Logout. Izgled stranice prikazan je na slici 4.5. Pritiskom na odgovarajuće dugme, aktiviraju se sljedeće funkcije.

| Naziv funkcije | Opis |
|---------------------------------|--|
| <i>users()</i> | Prebacuje korisnika na Users stranicu. |
| <i>devices()</i> | Prebacuje korisnika na Devices stranicu. |
| <i>package_channels()</i> | Prebacuje korisnika na PackagesChannels stranicu. |
| <i>check_admin_privileges()</i> | Šalje zahtjev na rutu zaduženu za provjeru da li je trenutni administrator, super administrator. Ukoliko jeste, prebacuje korisnika na Admin stranicu. |
| <i>logout()</i> | Vraća korisnika na početnu, Home, stranicu i briše pristupni token i token za obnovu. |

Tabela 4.11 Funkcije Menu stranice



Slika 4.5 Menu stranica

Users:

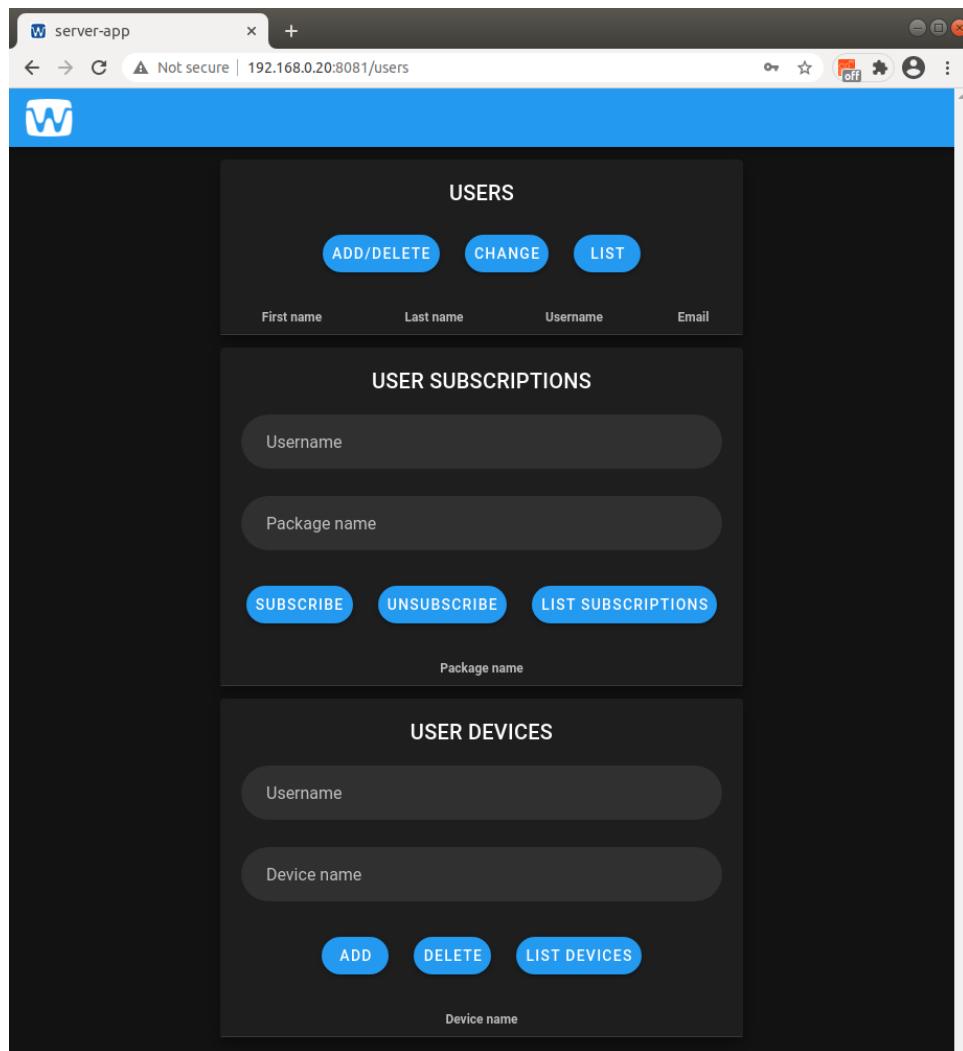
Stranica posjeduje tri kartice: Users (tri dugmeta Add/Delete, Change, List), User Subscriptions (dva polja za unos Username-a i Package name-a, tri dugmeta Subscribe, Unsubscribe, List Subscriptions), User Devices (dva polja za unos Username-a i Device name-a, tri dugmeta Add, Delete, List Devices). Izgled stranice prikazan je na slici 4.6.

Pritiskom na odgovarajuće dugme aktiviraju se sljedeće funkcije.

| Naziv funkcije | Opis |
|--------------------------|---|
| <i>change_user()</i> | Prebacuje korisnika na ChangeUser stranicu. |
| <i>add_delete_user()</i> | Prebacuje korisnika na AddDeleteUser stranicu. |
| <i>list_users()</i> | Šalje zahtjev, bez dodatnih parametara. Prima listu svih korisnika i upisuje u listu <i>all_users</i> , koja se prikazuje na stranici, u obliku tabele. |

| | |
|----------------------------------|--|
| <i>subscribe()</i> | Šalje zahtjev, sa parametrima <i>username</i> i <i>package_name</i> . |
| <i>unsubscribe()</i> | Šalje zahtjev, sa parametrima <i>username</i> i <i>package_name</i> . |
| <i>list_subscriptions()</i> | Šalje zahtjev, sa parametrom <i>username</i> . |
| <i>add_device_to_user()</i> | Šalje zahtjev, sa parametrima <i>username</i> i <i>device_name</i> . |
| <i>delete_device_from_user()</i> | Šalje zahtjev, sa parametrima <i>username</i> i <i>device_name</i> . |
| <i>list_user_devices()</i> | Šalje zahtjev, sa parametrom <i>username</i> . Prima listu svih uređaja korisnika i upisuje u listu <i>all_user_devices</i> , koja se prikazuje na stranici, u obliku tabele. |

Tabela 4.12 Funkcije User stranice



Slika 4.6 User stranica

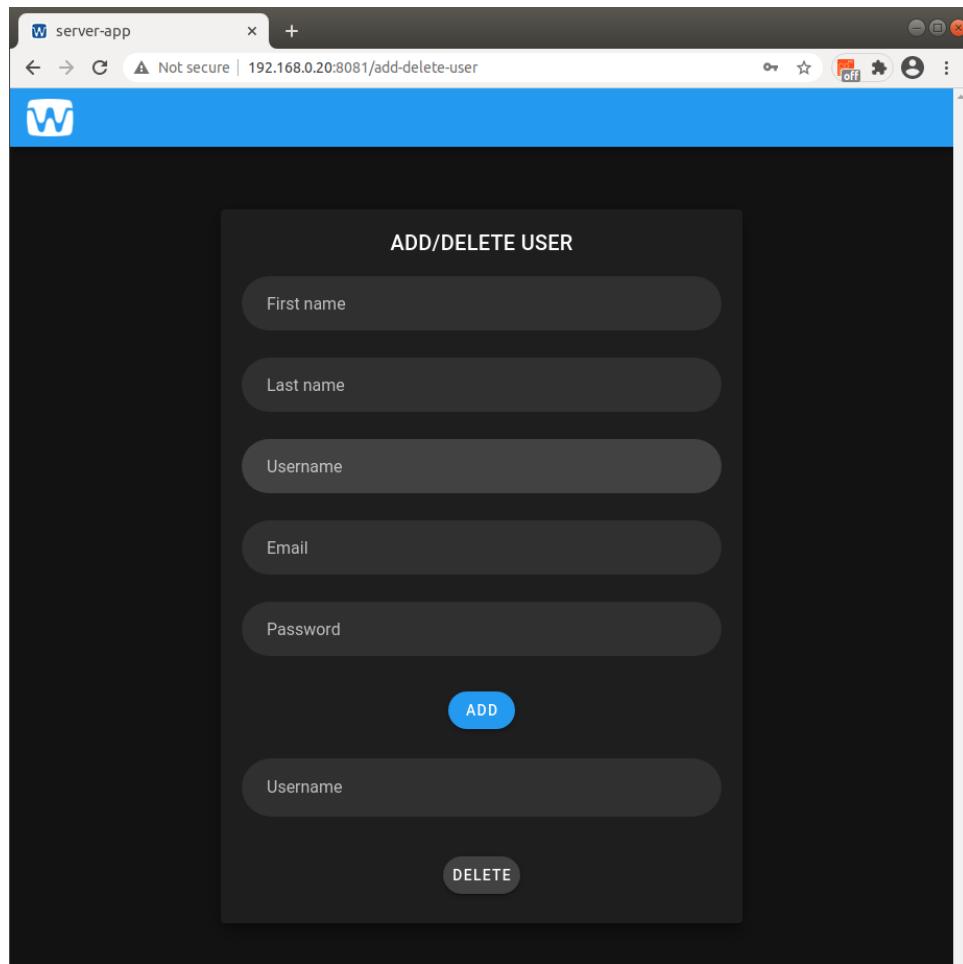
AddDeleteUser:

Stranica sadrži pet polja za unos teksta, u koje se upisuju podaci o korisniku, dugme Add. Takođe, sadrži jedno polje za unos teksta u koje se upisuje Username korisnika koji se briše i dugme Delete. Izgled stranice prikazan je na slici 4.7.

Pritiskom na dugme Add ili Delete, aktiviraju se odgovarajuće funkcije.

| Naziv funkcije | Opis |
|----------------------|--|
| <i>add_user()</i> | Šalje zahtjev, sa parametrima o korisniku: <i>first_name, last_name, username, email, password.</i> |
| <i>delete_user()</i> | Šalje zahtjev, sa parametrom <i>username</i> . |

Tabela 4.13 Funkcije AddDeleteUser stranice



Slika 4.7 AddDeleteUser stranica

ChangeUser:

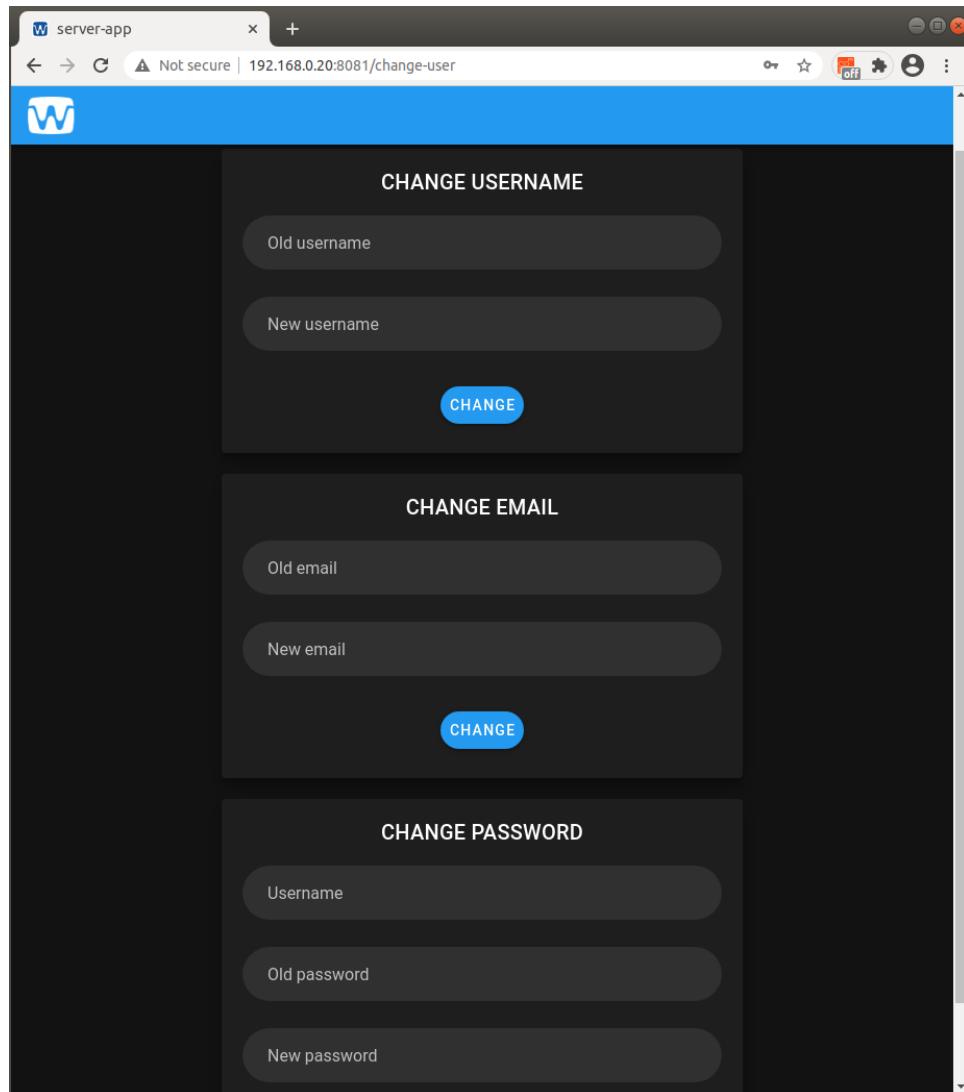
Stranica se sastoji od tri kartice, Change username (sadrži dva polja za unos teksta: Old username i New username) i dugme Change, Change email (sadrži dva polja za unos teksta: Old email, New email) i dugme Change, Change password (sadrži tri polja za unos teksta: Username, Old password, New password) i dugme Change. Izgled stranice prikazan je na slici 4.8.

Pritiskom na odgovarajuće dugme aktiviraju se sljedeće funkcije.

| Naziv funkcije | Opis |
|--------------------------------|--|
| <code>change_username()</code> | Šalje zahtjev, sa parametrima: <i>old_username</i> i <i>new_username</i> . |

| | |
|--------------------------|---|
| <i>change_email()</i> | Šalje zahtjev, sa parametrima: <i>old_email</i> i <i>new_email</i> . |
| <i>change_password()</i> | Šalje zahtjev sa parametrima: <i>username</i> , <i>old_password</i> , <i>new_password</i> . |

Tabela 4.14 Funkcije ChangeUser stranice



Slika 4.8 ChangeUser stranica

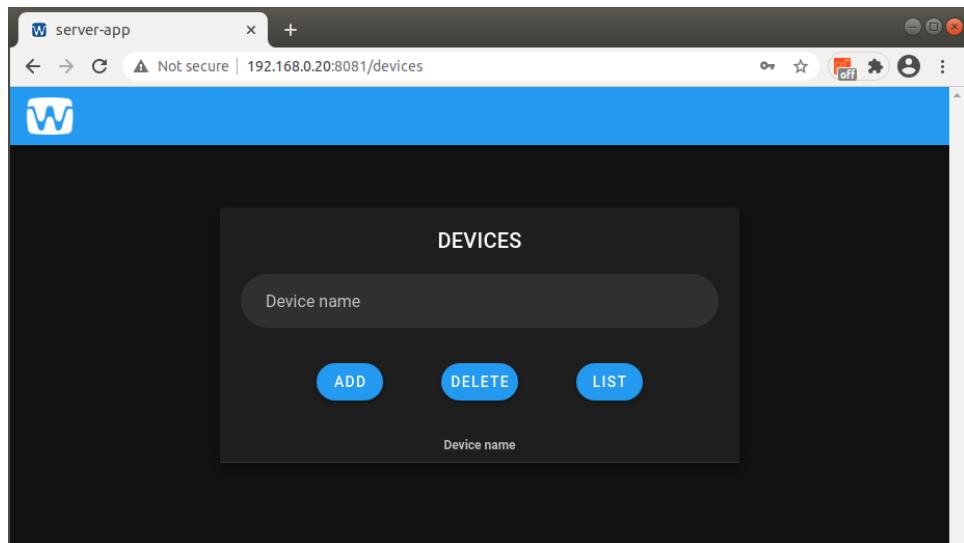
Devices:

Stranica posjeduje jedno polje za unos naziva uređaja (Device name) i tri dugmeta: Add, Delete, List. Izgled stranice prikazan je na slici 4.9.

Funkcije opisane u nastavku, se aktiviraju pritiskom na odgovarajuće dugme.

| Naziv funkcije | Opis |
|------------------------------|---|
| <code>add_device()</code> | Šalje zahtjev, sa parametrom <code>device_name</code> . |
| <code>delete_device()</code> | Šalje zahtjev, sa parametrom <code>device_name</code> . |
| <code>list_devices()</code> | Šalje zahtjev, bez dodatnih parametara. Prima listu svih uređaja i upisuje u listu <code>all_devices</code> , koja se prikazuje na stranici, u obliku tabele. |

Tabela 4.15 Funkcije Devices stranice



Slika 4.9 Devices stranica

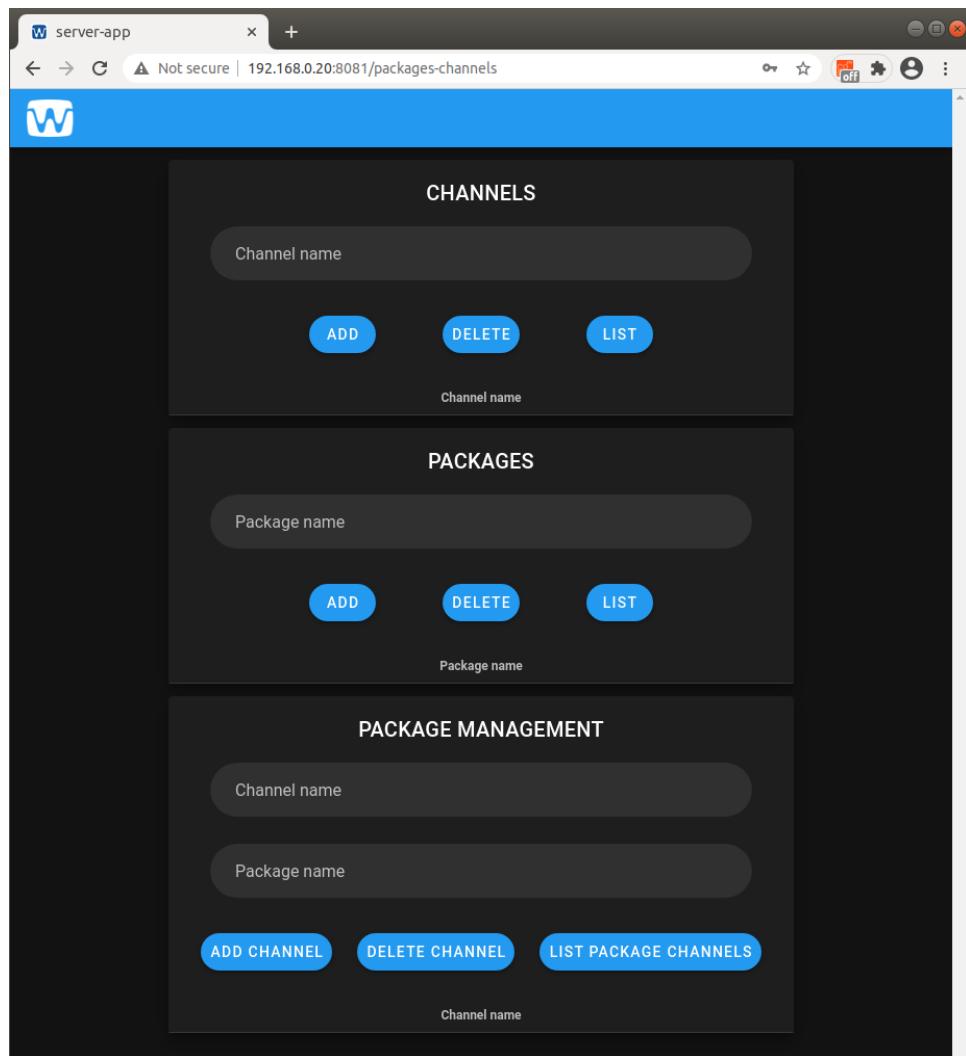
PackageChannels:

Stranica se sastoji iz tri kartic: Channels, Packages i Package management. Prva kartica sadrži polje za unos naziva kanala i tri dugmeta Add, Delete i List. Druga kartica sadrži polje za unos naziva paketa i tri dugmeta: Add, Delete, List. Treća kartica sadrži dva polja za unos, naziva kanala i naziva paketa, kao i tri dugmeta: Add channel, Delete channel, List package channels. Izgled stranice prikazan je na slici 4.10.

U nastavku su opisane funkcije, koje se aktiviraju pritiskom na odgovarajuće dugme.

| Naziv funkcije | Opis |
|--------------------------------------|---|
| <i>add_channel()</i> | Šalje zahtjev, sa parametrom <i>channel_name</i> . |
| <i>delete_channel()</i> | Šalje zahtjev, sa parametrom <i>channel_name</i> . |
| <i>list_channels()</i> | Šalje zahtjev, bez dodatnih parametara. Prima listu svih kanala i upisuje u listu <i>all_channels</i> , koja se prikazuje na stranici, u obliku tabele. |
| <i>add_package()</i> | Šalje zahtjev, sa parametrom <i>package_name</i> . |
| <i>delete_package()</i> | Šalje zahtjev, sa parametrom <i>package_name</i> . |
| <i>list_packages()</i> | Šalje zahtjev, bez dodatnih parametara. Prima listu svih paketa i upisuje u listu <i>all_packages</i> , koja se prikazuje na stranici, u obliku tabele. |
| <i>add_channel_to_package()</i> | Šalje zahtjev, sa parametrom <i>channel_name</i> i <i>package_name</i> . |
| <i>delete_channel_from_package()</i> | Šalje zahtjev, sa parametrom <i>channel_name</i> i <i>package_name</i> . |
| <i>list_package_channels()</i> | Šalje zahtjev, sa parametrom <i>package_name</i> . Prima listu svih kanala u paketu i upisuje u listu <i>all_package_channels</i> , koja se prikazuje na stranici, u obliku tabele. |

Tabela 4.16 Funkcije PackageChannels stranice



Slika 4.10 PackageChannels stranica

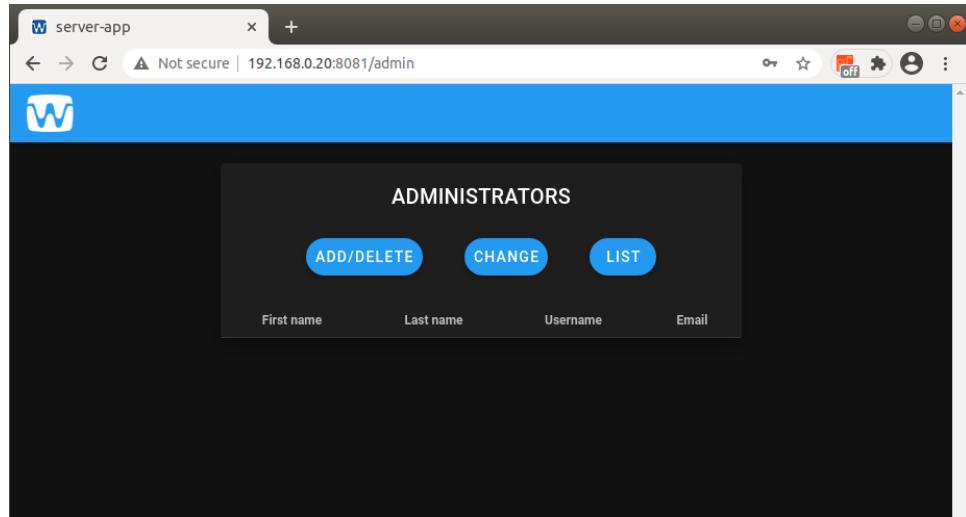
Admin:

Stranica sadrži tri dugmeta: Add/Delete, Change i List. Izgled stranice prikazan je na slici 4.11. Pritiskom na odgovarajuće dugme aktiviraju se funkcije koje će biti opisane u nastavku.

| Naziv funkcije | Opis |
|---------------------------|---|
| <i>add_delete_admin()</i> | Prebacuje korisnika na AddDeleteAdmin stranicu. |
| <i>change_admin()</i> | Prebacuje korisnika na ChangeAdmin stranicu. |

| | |
|----------------------------|--|
| <code>list_admins()</code> | Šalje zahtjev, bez dodatnih parametara. Prima listu svih administratora u sistem i upisuje u listu <code>all_admins</code> , koja se prikazuje na stranici, u obliku tabele. |
|----------------------------|--|

Tabela 4.17 Funkcije Admin stranice



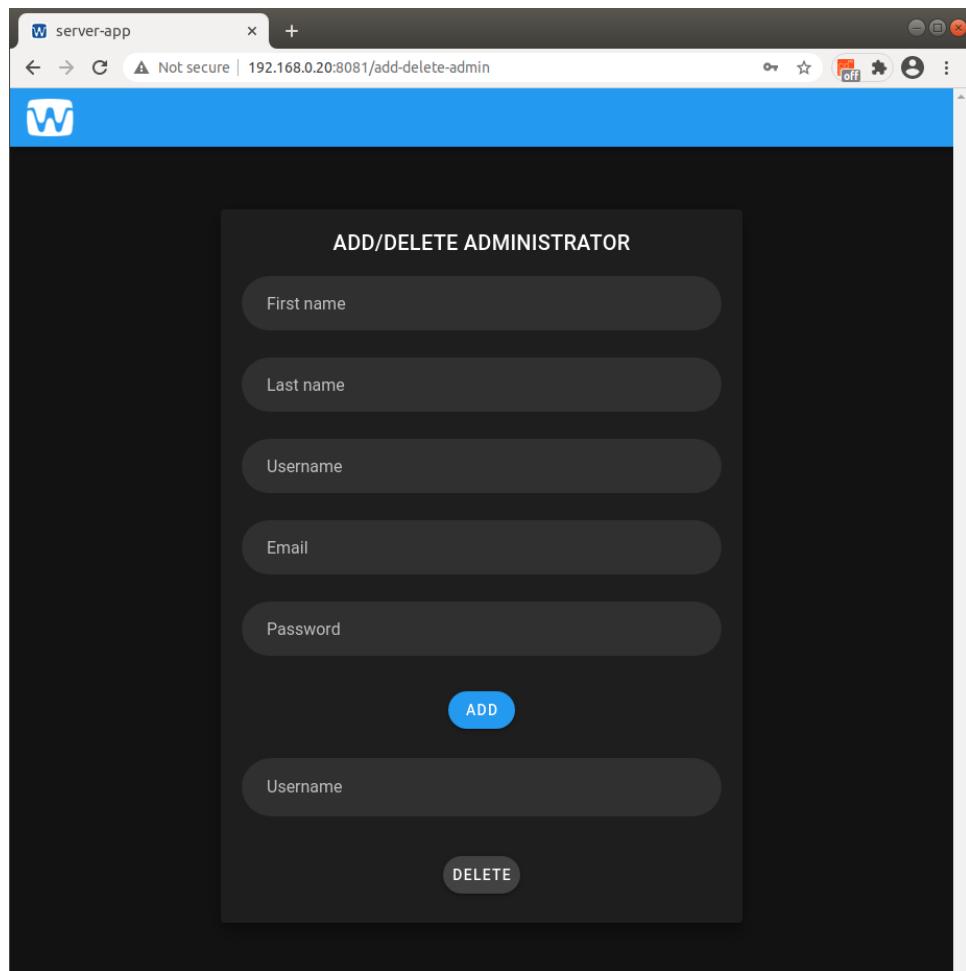
Slika 4.11 Admin stranica

AddDeleteAdmin:

Stranica se sastoje iz polja za upis teksta, u koje se unose podaci o administratoru koji se dodaje, polja za upis Username-a, administratora koji se briše i dva dugmeta Add i Delete. Izgled stranice prikazan je na slici 4.12. Pritisom na odgovarajuće dugme aktiviraju se sljedeće funkcije:

| Naziv funkcije | Opis |
|-----------------------------|--|
| <code>add_admin()</code> | Šalje zahtjev, sa parametrima: <code>first_name</code> , <code>last_name</code> , <code>username</code> , <code>email</code> , <code>password</code> . |
| <code>delete_admin()</code> | Šalje zahtjev, sa parametrom <code>username</code> . |

Tabela 4.18 Funkcije AddDeleteAdmin stranice



Slika 4.12 AddDeleteAdmin stranica

ChangeAdmin:

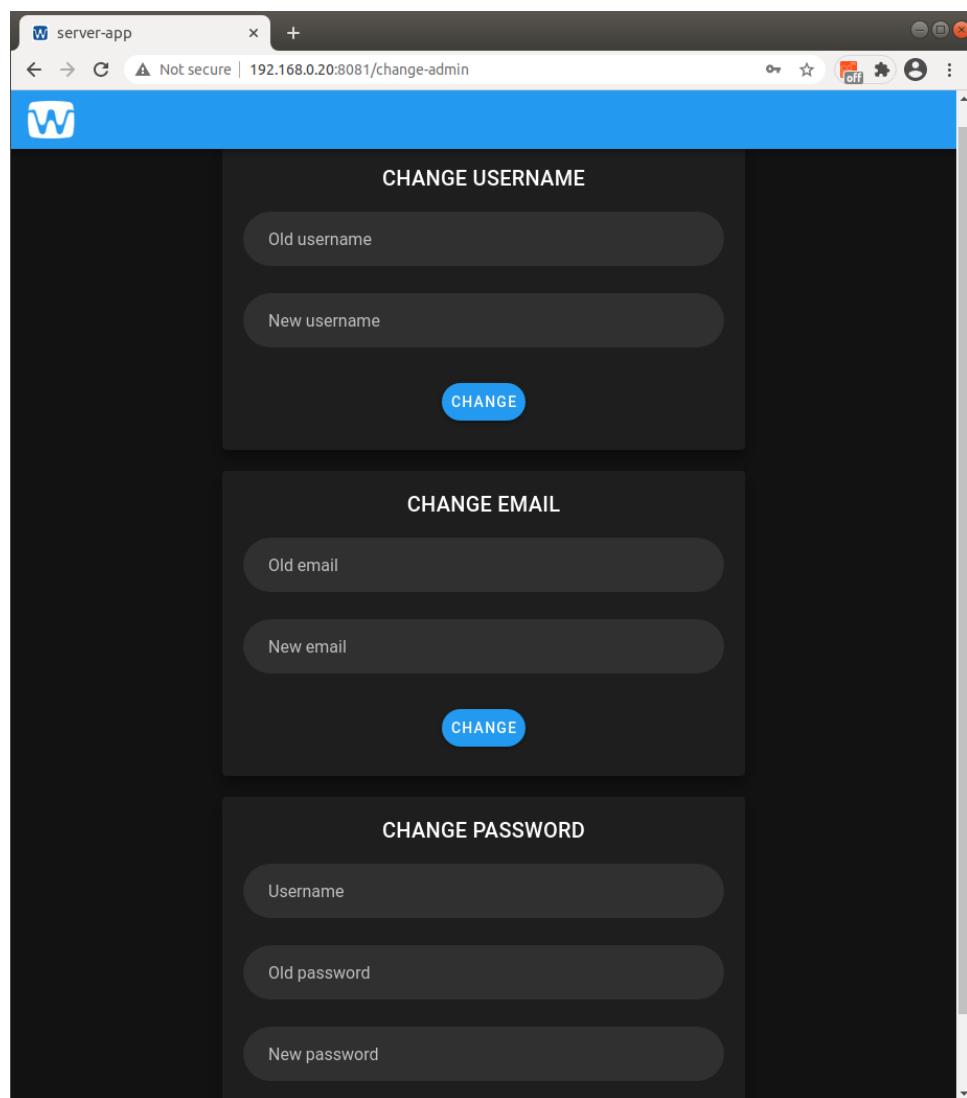
Stranica se sastoji od tri kartice: Change Username (sadrži dva polja za unos teksta, Old username, New username i dugme Change), Change Email (sadrži dva polja za unos teksta, Old email, New email i dugme Change), Change Password (sadrži tri polja za unos teksta, Username, Old password, New password i dugme Change). Izgled stranice prikazan je na slici 4.13.

Pritiskom na odgovarajuće dugme aktiviraju se funkcije koje će biti opisane u nastavku.

| Naziv funkcije | Opis |
|--------------------------|--|
| <i>change_username()</i> | Šalje zahtjev, sa parametrima: <i>old_username</i> i <i>new_username</i> . |
| <i>change_email()</i> | Šalje zahtjev sa parametrima: <i>old_email</i> i |

| | |
|--------------------------|--|
| | <i>new_email.</i> |
| <i>change_password()</i> | Šalje zahtjev sa parametrima: <i>username</i> , <i>old_password</i> i <i>new_password</i> . |

Tabela 4.19 Funkcije ChangeAdmin stranice



Slika 4.13 ChangeAdmin stranica

5. Ispitivanje realizovanog sistema

U ovom poglavlju, opisan je proces pokretanje sistema, nakon čega su dati testni primjeri svih funkcionalnosti opisanih u prethodnim poglavljima.

Serverska aplikacija se pokreće iz *PyCharm* IDE-a (*Integrated development environment*) i čeka na zahtjeve klijentske aplikacije na unaprijed definisanoj adresi i portu. Komandom *npm run serve*, pokreće se klijentska aplikacija, kojoj se pristupa unošenjem odgovarajuće adrese u web pretraživač.

U klijentskoj aplikaciji se unose potrebne informacije, pritišće se odgovarajuće dugme, nakon čega se šalje HTTP zahtjev ka serverskoj aplikaciji. Nakon obrade zahtjeva i izvršenja zadate operacije, serverska aplikacije vraća HTTP odgovor, koji se zatim obrađuje u klijentskoj aplikaciji, nakon čega se na ekranu ispisuje odgovarajuća notifikacija.

Serverska i klijentska aplikacija zajedno čine jedan zaokružen sistem, zbog čega je testiranje vršeno korištenjem klijentske aplikacije.

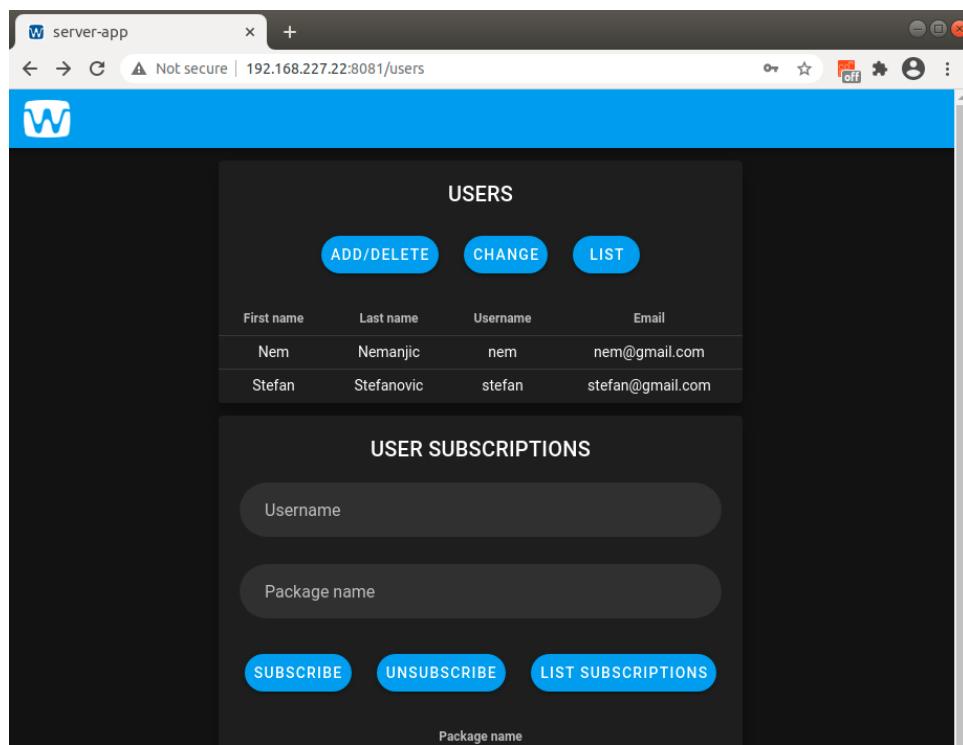
Testiranje je grupisano u četiri kategorije: upravljanje korisnicima, upravljanje uređajima, upravljanje paketima i upravljanje administratorima.

5.1 Ispitivanje funkcionalnosti upravljanja korisnicima

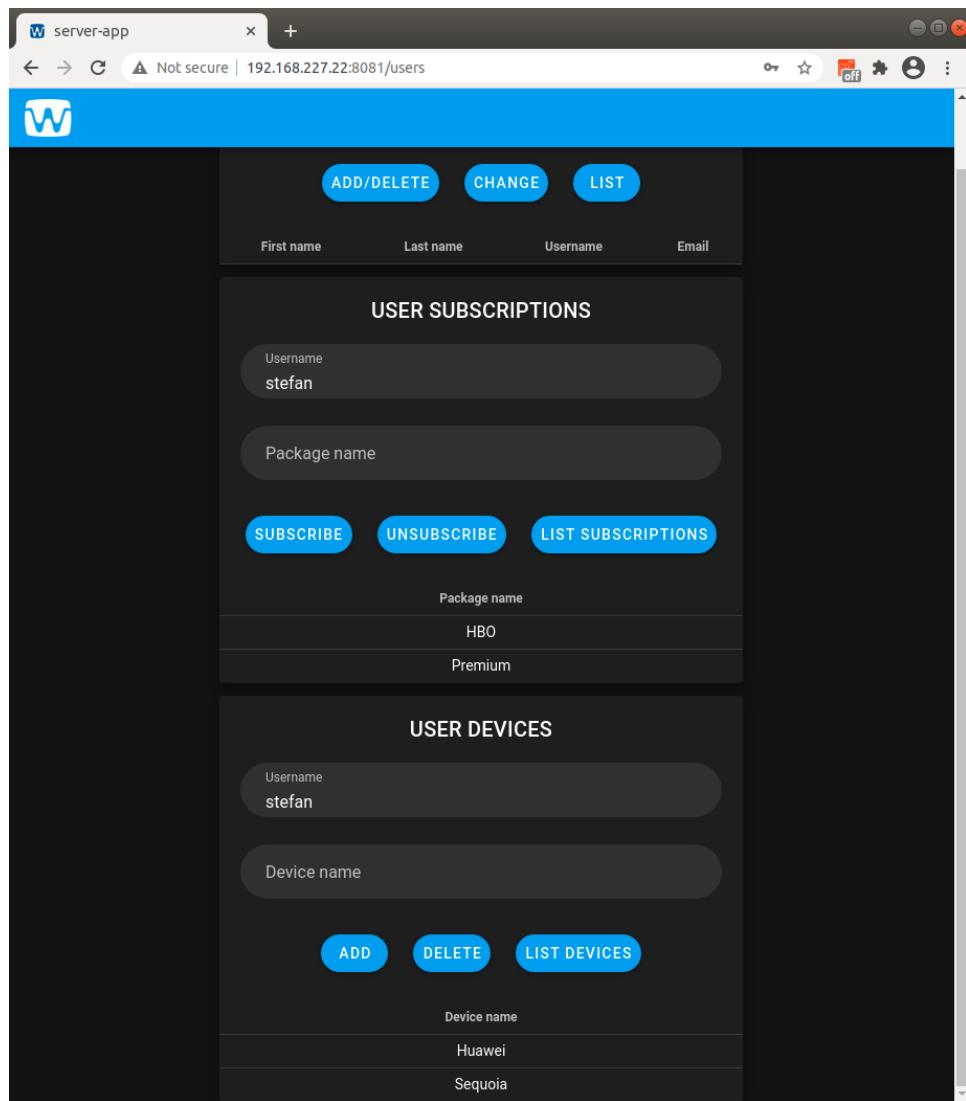
U nastavku je opisano testiranje svih funkcionalnosti vezanih za upravljanje korisnicima koji su prisutni u sistemu. Na slici 5.1 prikazan je izgled stranice nakon dodavanja dva nova korisnika i njihovog izlistavanja. Na slici 5.2 prikazan je izgled stranice nakon dodavanja novih uređaja korisniku i pretplaćivanja korisnika na nove pakete.

| Test | Rezultat |
|--|----------|
| Dodavanje novog korisnika. | Prošao |
| Uklanjanje postojećeg korisnika. | Prošao |
| Izlistavanje svih korisnika u sistemu. | Prošao |
| Promjena imejl adrese korisnika. | Prošao |
| Promjena nadimka korisnika. | Prošao |
| Promjena lozinke korisnika. | Prošao |
| Izlistavanje svih uređaja koji pripadaju jednom korisniku. | Prošao |
| Dodavanje novog uređaja korisniku. | Prošao |
| Brisanje uređaja koji posjeduje korisnik. | Prošao |
| Pretplaćivanje korisnika na novi paket. | Prošao |
| Brisanje pretplate korisnika sa paketa. | Prošao |
| Izlistavanje svih paketa na koje je korisnik pretplaćen. | Prošao |

Tabela 5.1 Ispitivanje funkcionalnosti upravljanja korisnicima



Slika 5.1 Upravljanje korisnicima - Izlistavanje korisnika



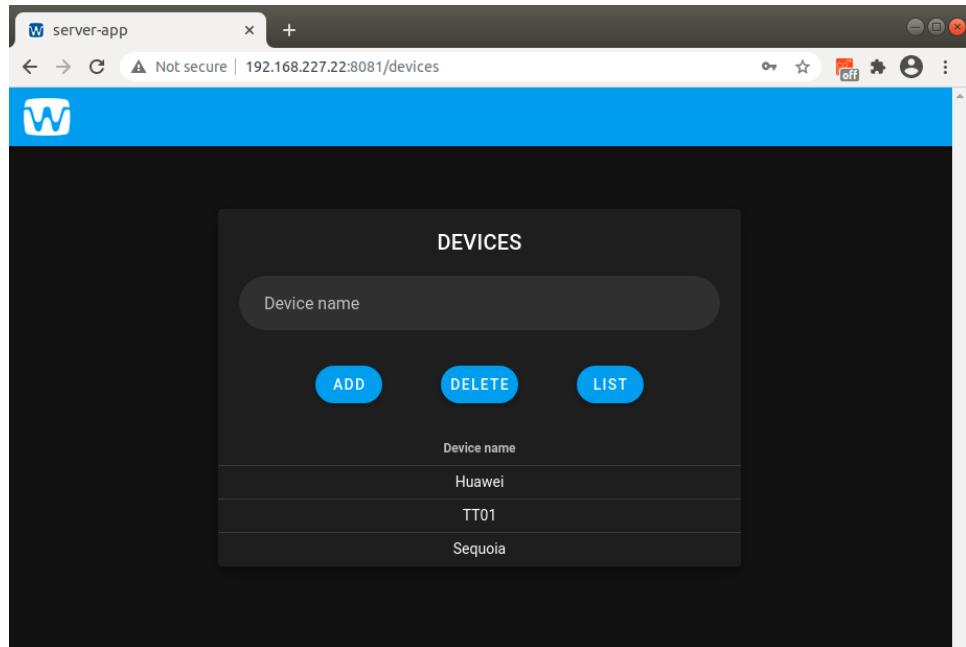
Slika 5.2 Upravljanje korisnicima - Izlistavanje korisničkih uređaja i pretplata

5.2 Ispitivanje funkcionalnosti upravljanja uređajima

U nastavku je opisano ispitivanje svih funkcionalnosti vezanih za upravljanje uređajima prisutnih u sistemu. Na slici 5.3 prikazan je izgled stranice nakon dodavanja novih uređaja u sistem i njihovog izlistavanja.

| Test | Rezultat |
|--------------------------------------|----------|
| Dodavanje novog uređaja. | Prošao |
| Brisanje postojećeg uređaja. | Prošao |
| Izlistavanje svih uređaja u sistemu. | Prošao |

Tabela 5.2 Ispitivanje funkcionalnosti upravljanja uređajima



Slika 5.3 Izlistavanje uređaja

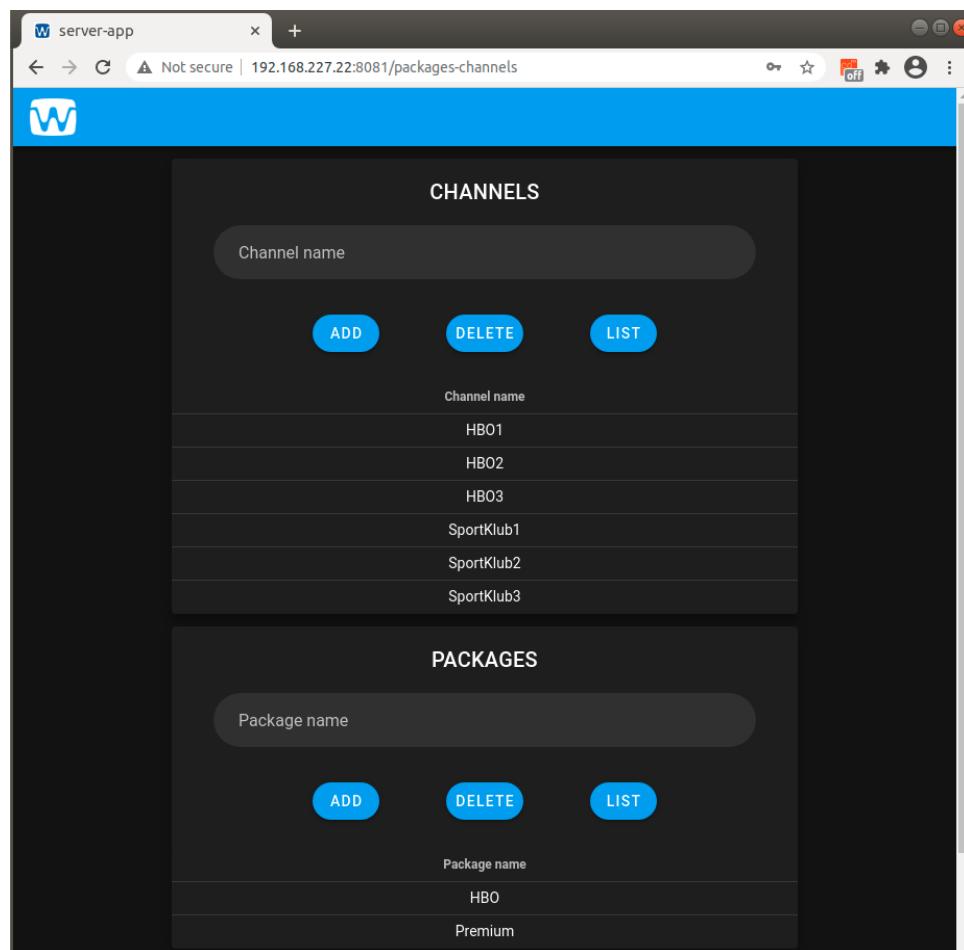
5.3 Ispitivanje funkcionalnosti upravljanja paketima i kanalima

U nastavku je opisano ispitivanje svih funkcionalnosti vezanih za upravljanje paketima i kanalima. Na slici 5.4 prikazan je izgled stranice nakon dodavanja novih kanala i paketa u sistem i njihovog izlistavanja.

| Test | Rezultat |
|---|----------|
| Dodavanje novog kanala. | Prošao |
| Brisanje postojećeg kanala. | Prošao |
| Izlistavanje svih kanala prisutnih u sistemu. | Prošao |
| Dodavanje novog paketa. | Prošao |
| Brisanje postojećeg paketa. | Prošao |

| | |
|---|--------|
| Izlistavanje svih paketa prisutnih u sistemu. | Prošao |
| Dodavanje kanala u paket. | Prošao |
| Brisanje kanala iz paketa. | Prošao |
| Izlistavanje svih kanala prisutnih u paketa. | Prošao |

Tabela 5.3 Ispitivanje funkcionalnosti upravljanja paketima i kanalima



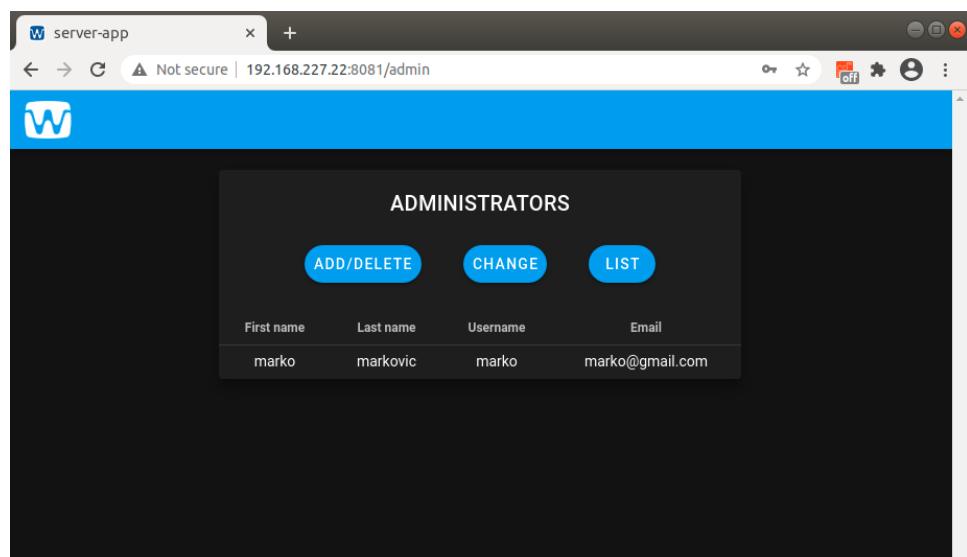
Slika 5.4 Izlistavanje kanala i paketa u sistemu

5.4 Ispitivanje funkcionalnosti upravljanja administratorima

U nastavku je opisano ispitivanje svih funkcionalnosti vezanih za upravljanje administratorima u sistemu. Na slici 5.5 prikazan je izgled stranice nakon dodavanja novih administratora u sistem i njihovog izlistavanja.

| Test | Rezultat |
|---|----------|
| Registracija administratora. | Prošao |
| Prijavljivanje administratora. | Prošao |
| Dodavanje novog administratora. | Prošao |
| Brisanje postojećeg administratora. | Prošao |
| Izlistavanje svih administratora prisutnih u sistemu. | Prošao |
| Promjena imejl adrese administratora. | Prošao |
| Promjena nadimka administratora. | Prošao |
| Promjena lozinke administratora. | Prošao |
| Provjera privilegije administratora. | Prošao |

Tabela 5.4 Ispitivanje funkcionalnosti upravljanja administratorima



Slika 5.5 Izlistavanje administratora

6. Zaključak

U ovom radu je opisano jedno rješenje, u vidu full stack aplikacije, namijenjene administratorima DTV provajdera, sa ciljem kreiranja jednog sveobuhvatnog sistema, sa kojim će provajderi usluga moći da upravljaju korisnicima, uređajima i sadržajem, unutar svog sistema.

U web pretraživač se unosi adresa web aplikacije, nakon čega se administrator registruje, a zatim i prijavljuje. Zahtjevi za registraciju i prijavljivanje, kao i svi ostali zahtjevi, koriste HTTP protokol i JSON format za razmjenu podataka i šalju zahtjev ka serverskoj aplikaciji, koja vrši obradu zahtjeva i vraća odgovarajući odgovor. Nakon uspješnog prijavljivanja administratora, dobija se pristup ostatku web aplikacije, putem koje administrator može da dodaje nove korisnike, briše postojeće, mijenja njihove podatke, dodaje i briše uređaje i pakete koje posjeduju korisnici, dodaje i briše kanale pakete i uređaje koji su prisutni u sistemu. Takođe, ukoliko administrator posjeduje privilegije super administratora, on može da dodaje nove i briše postojeće administratore i da mijenja njihove podatke.

Jedna od prednosti realizovanog sistema je njegova sveobuhvatnost. Administratori kroz jedinstvenu aplikaciju vrše manipulisanje korisnicima, uređajima i sadržajem u sistemu i veoma lako dodjeljuju korisniku nove uređaje ili pakete. Akcenat ovog rada bila je serverska aplikacija, stoga su mane prvenstveno vezane sa web aplikaciju. Aplikacija nije prilagođena za prikazivanje na mobilnim uređajima i tabletima, nego samo na računarima, što u današnje vrijeme predstavlja manu koja bi u slučaju komercijalizacije aplikacije morala biti riješena. Još jedna mana i moguće unapređenje bi bilo veće korištenje komponenti unutar jedne web stranice, umjesto korištenja više stranica, što bi iskoristilo potencijale *VueJS* radnog okvora.

7. Literatura

- [1] Istorijat televizije, dostupno na: <https://www.britannica.com/technology/television-technology>, pristupano februar 2021.
- [2] dr Milan Z. Bjelica, dr Nikola Teslić, mr Velibor Mihić, Softver u digitalnoj televiziji 1, Univerzitet u Novom Sadu, Fakultet Tehničkih Nauka, 2017.
- [3] API, dostupno: <https://www.pcmag.com/encyclopedia/term/api>, pristupano februar 2021.
- [4] REST, dostupno na: <https://www.smashingmagazine.com/2018/01/understanding-using-rest-api/>, pristupano februar 2021.
- [5] Hypertext Transfer Protocol – HTTP/1.1., R.T. Fielding, J. Gettys, J.C. Mogul et al., RFC 2616, The IETF Trust, 1999.
- [6] JSON, dostupno na: <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON>, pristupano februar 2021.
- [7] JSON Web Token (JWT), M. Jones, J. Bradley, N. Sakimura, RFC 7519, IETF Trust, Maj 2015.
- [8] Python, dostupno na: <https://pythoninstitute.org/what-is-python/>, pristupano februar 2021.
- [9] SQLite, dostupno na:
<https://ijcsmc.com/docs/papers/April2015/V4I4201599a35.pdf>, pristupano februar 2021.