Platforma za daljinski nadzor i upravljanje uređajima

Klišković, Kristian

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

PLATFORMA ZA DALJINSKI NADZOR I UPRAVLJANJE UREĐAJIMA

Diplomski rad

Kristian Klišković

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 20.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Kristian Klišković
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. Pristupnika, godina upisa:	D-1369, 07.10.2021.
OIB studenta:	91030832716
Mentor:	prof. dr. sc. Josip Job
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Ratko Grbić
Član Povjerenstva 1:	prof. dr. sc. Josip Job
Član Povjerenstva 2:	Robert Šojo, mag. ing. comp.
Naslov diplomskog rada:	Platforma za daljinski nadzor i upravljanje uređajima
Znanstvena grana diplomskog rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Zadatak ovog diplomskog rada je proučiti postojeće sustave za prikupljanje podataka i upravljanje udaljenim uređajima te predložiti dizajn sustava koji će korisnicima omogučiti proširivost putem API-ja, neovisno o vrsti koristenog hardweara. Predloženo rješenje je potrebno implementirati te demonstrirati i analizirati njegove mogućnosti. Tema rezervirana za: Kristian Klišković
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	20.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 15.12.2023.

Ime i prezime studenta:	Kristian Klišković
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D-1369, 07.10.2021.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Platforma za daljinski nadzor i upravljanje uređajima**

izrađen pod vodstvom mentora prof. dr. sc. Josip Job

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. Uvod

U današnjem digitalnom dobu, internet stvari (IoT) postaje ključna komponenta u razvoju tehnoloških rješenja koja povezuju fizičke objekte s digitalnim svijetom. IoT pruža priliku za stvaranje inteligentnih sustava koji poboljšavaju kvalitetu života i optimiziraju operativne procese u različitim sektorima, uključujući industriju, energetiku, zdravstvo, promet i mnoge druge. Centralna uloga u razvoju i implementaciji IoT rješenja pripada platformama za upravljanje mikro kontrolerima koje omogućavaju integraciju, nadzor i kontrolu IoT uređaja.

Ovaj diplomski rad prikazuje implementaciju platforme za nadzor i upravljanje uređajima. Cilj platforme je omogućiti korisnicima nadzor i upravljanje različitih uređaja putem mobilne aplikacije.

U drugom poglavlju ovog diplomskog rada, pružit ćemo površni pregled funkcionalnosti platforma MyDevices Cayenne i Blynk. Ovdje ću istaknuti nekoliko ključnih funkcija svake od ovih platformi, dajući osnovni uvid u ono što svaka platforma može pružiti.

U trećem poglavlju objasnit ću svoju implementaciju platforme za nadzor i upravljanje uređajima. Bit će opisane sve funkcionalnosti poslužitelja, kako koristiti korisničko sučelje te kako napraviti vlastiti uređaj kompatibilan sa sustavom.

Četvrto poglavlje je posvećeno testiranju platforme. Izrađeno je nekoliko fizičkih instanci uređaja kojima bi se upravljalo i koje su korištene za testiranje svih implementiranih značajki platforme. Također su navedeni neki zanimljivi primjeri korištenja platforme. Istaknute su i poželjne karakteristike koje nisu uključene, a koje se mogu pronaći u komercijalnim platformama za nadzor i upravljanje uređajima.

2. Komercijalno dostupne platforme za nadzor i upravljanje uređajima

Na tržištu je dostupan veliki broj platformi i aplikacija za nadzor i upravljanje uređajima. U nastavku ću pobliže proučiti nekoliko njih.

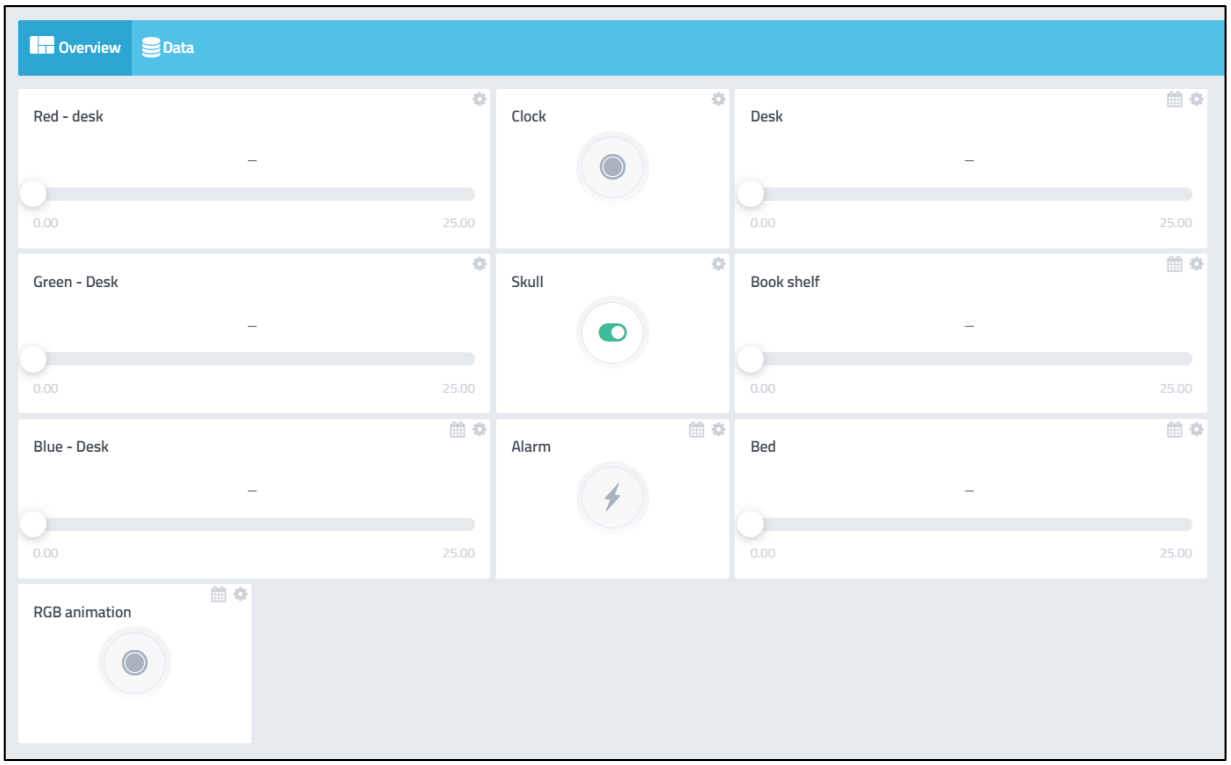
2.1. MyDevices Cayenne

MyDevices Cayenne je platforma za upravljanje IoT uređajima. Usluga je besplatna i jednostavna za korištenje. Sastoji se od servera te android i web aplikacije. Dok Web aplikacija radi poprilično dobro android aplikacija nije više dostupna na google play trgovini no može se pronaći njena .apk instalacijska datoteka. Android aplikacija ima nekoliko velikih propusta što se tiče dizajna i osnovnih funkcionalnosti. Svaki puta kada se želi koristiti aplikacija, mora se upaliti, ugasiti pa ponovno upaliti kako bi sve funkcionalnosti ispravno radile. MyDevices platforma ima široku dokumentaciju koja uvelike pomaže pri postavljanju sustava [1].

MyDevices podržava nekoliko različitih uređaja: Raspberry Pi računala te Arduino i ESP mikrokontrolere. Komunikacija servera sa uređajem se ostvaruje preko MQTT protokola. Sustav podržava 14 vrsta senzora i 6 vrsta aktuatora za koje već ima pripremljen kod. Osim njih korisnik može dodati generičke MQQT (slanje generične brojčane vrijednosti između uređaja i poslužitelja). Način dodavanja uređaja i senzora ili aktuatora je poprilično jednostavan i prikazan je na slici 2.1.

Slika 2.1. Sučelje za dodavanje uređaja i komponenti u MyDevice platformi

Web aplikacija za MyDevices služi za administraciju uređaja i upravljanje uređajem. Primjer sučelja za upravljanje uređajem je prikazan na slici 2.2.

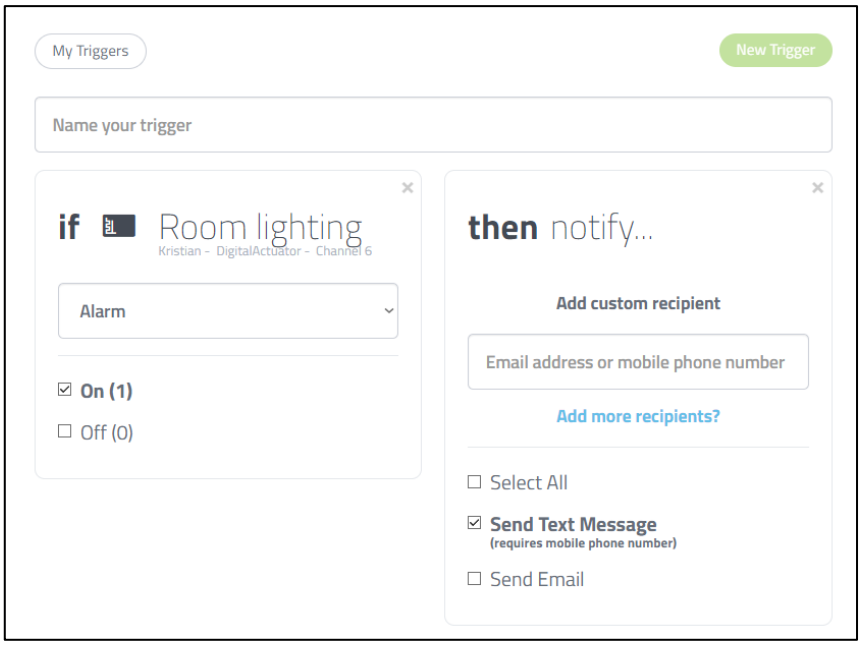


Slika 2.2. Primjer sučelja za upravljanje uređajem sa MyDevices platformom

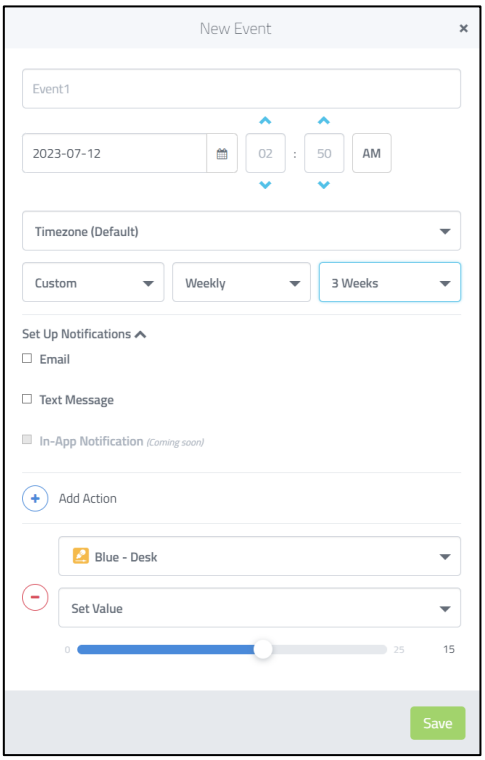
Komponente uređaja mogu biti dio jednog ili više projekata. Nakon što je projekt kreiran moguće je metodom „drag and drop“ dodavati komponente različitih uređaja u taj projekt. To je korisna funkcionalnost jer je moguće odvojiti sve istovrsne komponente, pregledavati ih i upravljati njima istovremeno bez prebacivanja uređaja.

MyDevices ima i dvije mogućnosti automatizacije: okidači i događaji. Okidači služe sa slanje obavijesti u obliku SMS poruke ili email-a u slučaju ispunjena nekakvog uvjeta. Uvjet može biti promjena dostupnosti uređaja (online – offline) ili da je vrijednost neke komponente dostigla željenu vrijednost. Izgled sučelja za dodavanje okidača je prikazan na slici 2.3.

Događaji postavljaju vrijednost neke komponente te su vremenski kontrolirani. Izgled sučelja za dodavanje događaja prikazan je na slici 2.4. Da bi se dodao događaj potrebno je postaviti sve njegove parametre koji su datum i vrijeme prvog okidanja, interval ponavljanja te koje sve radnje želimo da poduzme kada se ostvari vremenski uvjet.



Slika 2.3. Izgled sučelja za dodavanje okidača u MyDevices platformi



Slika 2.4. Izgled sučelja za dodavanje događaja u MyDevices platformi

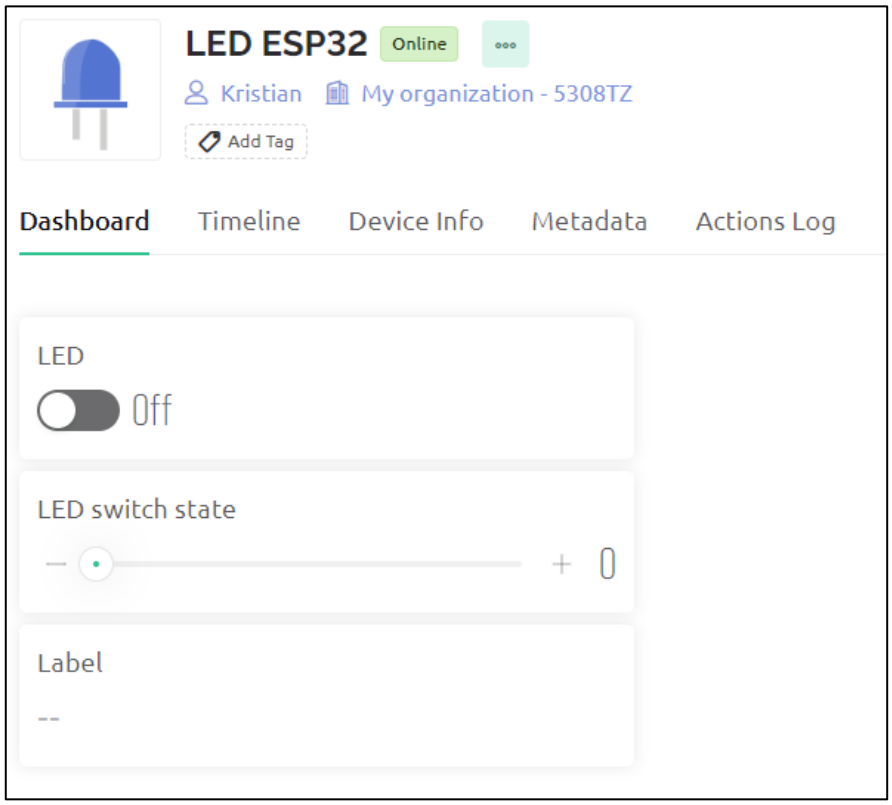
Pozitivne stvari o MyDevices platformi su besplatno korištenje, jednostavnost postavljanja i upotrebe, velika količina predgeneriranog programskog koda.

Negativne stvari su loša android aplikacija te sigurnosni propusti sustava. Kroz korištenje MyDevices sustava dulje od godine dana primijetio sam da stranica više puta nije radila i/ili bila izvor različitih računalnih virusa.

2.2. Blynk

Blynk je dosta modernija platforma za nadzor i upravljanje uređajima te nudi neke bolje funkcionalnosti. Podržava jako velik broj mikro kontrolera. Podržava puno načina upravljanja sa uređajem poput numeričkog klizača, analognog prikaza mjerene veličine, prikaza podataka na geografskoj karti, slanje slika, grafički prikaz podataka, višestruki izbor veličine i razne druge, ali jedini načini koji su dostupni u besplatnoj verziji su numerički klizač, tipka i tekstualni prikaz podataka. Ostale tipovi podataka su dostupni tek u pro verziji koja košta $499 godišnje. Više informacija o Blynk platformi je dostupno u Blynk dokumentaciji [2].

Prikaz konzole za upravljanje uređajima je prikazan na slici 2.5.

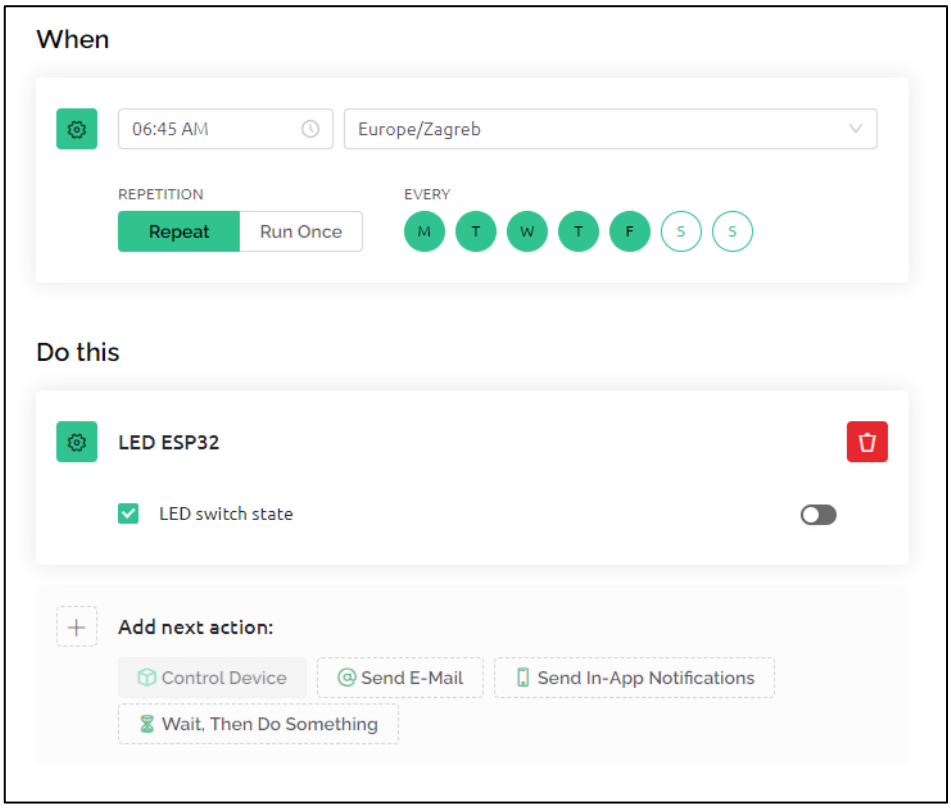


Slika 2.6. Prikaz sučelja za upravljanje uređajima u Blynk platformi

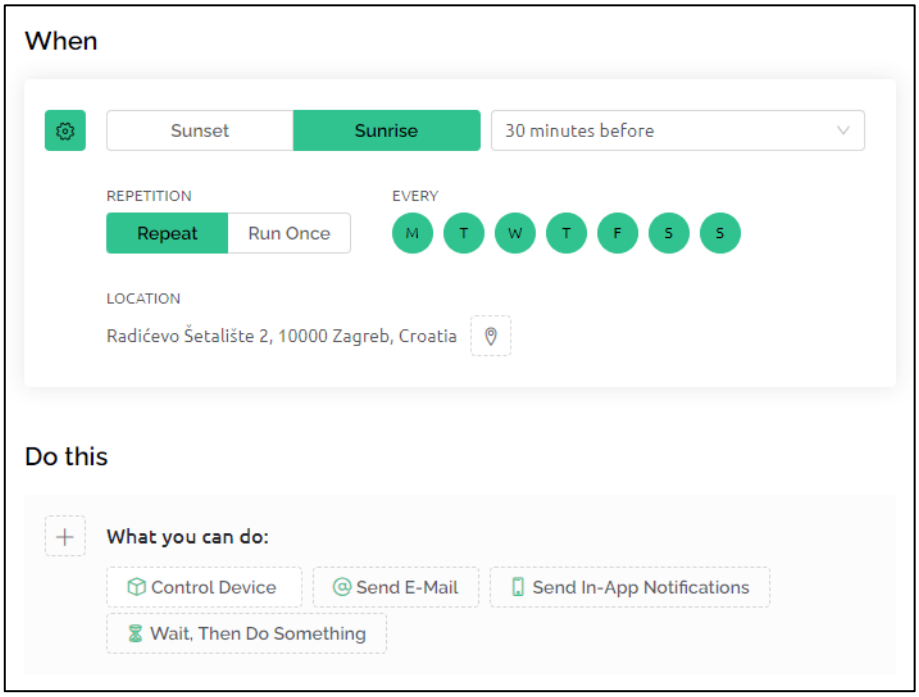
Blynk nudi dodavanje više korisnika na jedan uređaj. Svaki novi korisnik može imati jednu od 3 uloge: admin, osoblje ili obični korisnik. U besplatnoj verziji svaka vrsta korisnika ima fiksna prava, dok se u pro verziji može odrediti kakva sva prava imaju pripadnici svake skupine.

Blynk nudi 2 vrste okidača: vremenski okidači i okidači po stanju uređaja. Vremenski okidači se postavljaju na način da se odabere vrijeme okidanja te način ponavljanja odabiranjem dana u tjednu kada se treba okidati. Rezultat okidača može biti nekoliko različitih stavaka: slanje emaila, slanje mobilne notifikacije, mijenjanje stanja uređaja na neki način te se može dodati vremensko

odgađanje tih rezultata. Prikaz dodavanja novog vremenskog okidača je prikazan na slici 2.7. Vremenski okidač također može biti na izlazak ili zalazak sunca što je prikazano na slici 2.8.



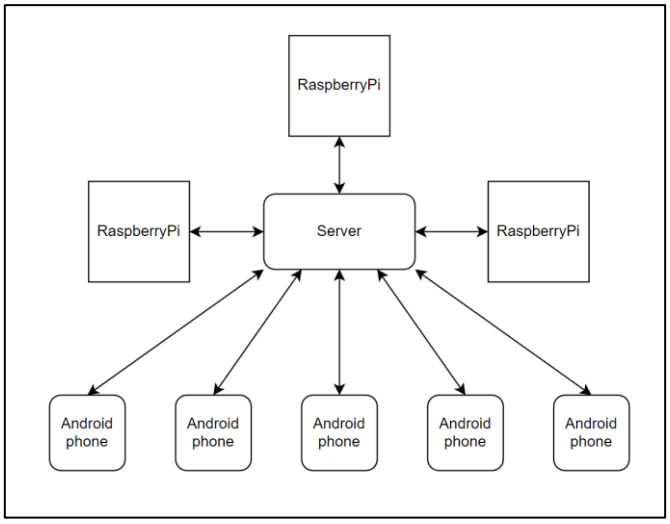
Slika 2.7. Prikaz dodavanja vremenskog okidača na Blynk platformi



Slika 2.8. Prikaz dodavanja vremenskog okidača sa izlaskom i zalaskom sunca

3. Platforma za daljinski nadzor i upravljanje uređajima

Razvijena platforma se sastoji od 3 komponente: Poslužitelj, korisničko sučelje te uređaji. Glavna komponenata je poslužitelj koji povezuje preostala dva dijela te im nudi razne usluge. Korisničko sučelje je android aplikacija pomoću koje korisnik može upravljati sustavom. Posljednji dio su uređaji kojima se upravlja odnosno koje se nadzire. U ovom poglavlju će se detaljno obrazložiti svaka komponenta. Struktura platforme je skicirana na slici 3.1. , na slici je za primjer naveden RaspberryPi kao uređaj.



Slika 3.1. Struktura platforme

Zahtjevi koje platforma mora zadovoljiti su:

1) Dinamički broj uređaja i korisnika u sustavu, odnosno mora postojati način registracije i brisanja korisnika i uređaja.
2) Više načina upravljanja stanjem uređaja, omogućiti korisniku da na različite načine upravlja različitim komponentama – da sučelje za upravljanje RGB trakom bude drugačije od sučelja za upravljanje relejom.
3) Brz odziv sustava na promjene stanja, kada korisnik promjeni stanje uređaja kroz aplikaciju, promjena se treba reflektirati na uređaju kroz maksimalno 5 sekundi.
4) Omogućavanje vlasniku uređaja da podjeli dopuštenja drugim korisnicama na određene komponente uređaja.
5) Okidači – za ispunjen zadani uvjet odraditi određenu akciju. Uvjet može biti određeno stanje uređaja ili vremenski uvjet, a akcija može biti nekakva notifikacija ili promjena stanja uređaja.

3.1. Arhitektura uređaja

U ovom potpoglavlju objašnjena je podatkovna struktura koju uređaji moraju slijediti kako bi bio kompatibilni sa sustavom.

3.1.1. Vrste podataka

Uređaj razmjenjuje informacije preko polja podataka. Vrijednost svih polja uređaja definira ukupno trenutno stanje uređaja. Polje može biti jednog od pet vrsta: brojčano, tekstualno, boolean, polje sa višestrukim izborom i RGB polje. S tih pet vrsta je pokrivena svaka vrsta osnovne kontrole koje korisnik može zahtijevati.

Brojčano polje se definira pomoću minimalne i maksimalne vrijednosti polja, minimalnog koraka vrijednosti te tekstualnog prefiksa i sufiksa. Primjer korištenja brojčanog polja je kontrola svjetline rasvjete ili pregled mjerene veličine nekog senzora.

Boolean polje može biti u dva stanja – upaljeno i ugašeno, Može se modelirati boolean (binarnom) varijablom. Primjeri upotrebe boolean polja su paljenje/gašenje nekog releja ili praćenje stanja senzora s binarnim izlazom.

Polje sa višestrukim izborom definira se navođenjem nekoliko stanja primjerice: „Stanje1“, „Stanje2“, „Stanje3“. Tipična primjena polja sa višestrukim izborom je izbor boje LED trake između nekoliko predefiniranih vrijednosti.

RGB polje sastoji se od tri vrijednosti u rasponu od 0 do 255, koje predstavljaju heksadecimalni kod boje. Tipična primjena RGB polja uključuje odabir boje za LED traku.

Tekstualnim poljem se upravlja tekstom. Služi da se pokriju slučajevi u kojima nisu prigodna prethodno opisana polja.

Osim podataka koji specificiraju tip i ključne vrijednosti polja, svako polje ima i identifikacijski broj polja, ime polja i smjer toka podataka gdje ono može biti ulazno-izlazno polje ili izlazno polje. Izlazno polje je ono koje nije namijenjeno za manipulaciju od strane korisnika, primjerice stanje senzora temperature. Drugim riječima, samo uređaj može utjecati na stanje tog polja. Ulazno-izlaznom polju se može mijenjati vrijednost i sa strane korisnika i sa strane samog uređaja recimo polje za kontrolu svjetline LED trake.

3.1.2. Grupiranje podataka

Radi jednostavnijeg prikaza moguće je da korisnik zahtjeva grupiranje podataka. Polja se grupiraju u grupe i kompleksne grupe.

Grupa je jednostavna vrsta grupiranja jer se praktički sastoji samo od liste polja. Grupa se definira sa identifikacijskim brojem grupe, nazivom grupe te listom polja u toj grupi.

Kompleksna grupa ima stanja. Stanje kompleksne grupe se definira identifikacijskim brojem stanja, imenom tog stanja i listom polja u tom stanju. Kompleksnu grupu definiramo identifikacijskim brojem kompleksne grupe, nazivom kompleksne grupe te listom stanja.

Ako uređaj na sebi ima RGB LED traku, njome se može upravljati na više načina, recimo upravljanje svjetlinom trake ili odabirom točne nijanse boje. Kompleksna grupa omogućava korisniku kontrolu određenih izlaza uređaja na više načina ovisno o stanju kompleksne grupe. Za primjer RGB LED trake možemo postaviti da je jedno stanje kompleksne grupe ima naziv „Svjetlina“ te ima jedno brojčano polje, a drugo stanje ima naziv „Nijansa boje“ te se sastoji od jednog RGB polja. Na taj način bi osigurali više načina upravljanja istim izlazom.

Uređaj se sastoji od liste grupa, liste kompleksnih grupa, naziva uređaja, identifikacijskoj broja uređaja te tajnog ključa uređaja. Tajni ključ uređaja mora biti jedinstven za svaki uređaj. Bez tog ključa uređaj se nema kako identificirati poslužitelju jer jedini primarni ključ koji preostaje je identifikacijski broj uređaja koji se u iznimnim slučajevima može mijenjati između sesija, a nije ni pogodan za autentikaciju.

3.2. Poslužitelj

Poslužitelj je implementirana u Node.js [3] okruženju koristeći TypeScript i Express.js. Za pohranu podataka koristi se Firestore [4] baza podataka od Firebase-a.

Poslužitelj komunicira s korisnikom putem mobilnih notifikacija, e-mail obavijesti i android aplikacije putem različitih HTTP zahtjeva i websocket veza. Također, poslužitelj komunicira s uređajem putem HTTP zahtjeva i websocket veza.

U ovakvim sustavima, jako je velik naglasak na instantnost komunikacije poslužitelja sa svim uređajima i korisnicima, te se stoga koriste websocket veze pomoću kojih poslužitelj u svakom trenutku može poslati poruku svim korisnicima ili uređajima koji su spojeni na njega.

3.2.1. Autentikacija korisnika

Jedno od osnovnih komponenti svakog poslužitelja je autentikacija korisnika. Autentikacija je omogućena preko nekoliko HTTP poziva koji služe za registraciju, prijavu i odjavljivanje. Za sve autentikacije osim registracije i prijave sa lozinkom su korišteni tokeni, te se lozinka šalje u samo

u tim slučajevima. Omogućena je i prijava sa tokenom kako se na korisničkoj strani ne bi trebala pohranjivati lozinka.

Za registraciju korisnika potrebno je napraviti HTTP zahtjev opisan na slici 3.2.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/userAuth/register

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "username": "Kristian",
    "password": "lozinka",
    "email": "kristiankliskovic@student.ferit.hr",
    "firebaseToken": "jssdf-5346-reterterzt"
}
```

Slika 3.2. Opis HTTP zahtjeva za registraciju korisnika

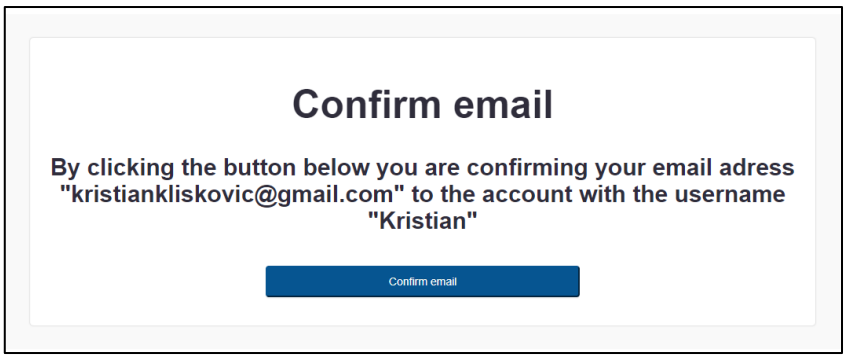
Ukoliko već postoji korisnik s tim imenom vratiti će se negativan odgovor (HTTP status kod 400), no ako je moguće uspješno kreirati korisnički račun vratit će se odgovor strukture prema slici 3.3.

```
HTTP status kod: 200 OK

Tijelo odgovora:
{
    "id": 10,
    "email": "",
    "username": "Kristian",
    "authToken": "4353-fdjk-primjerTokena"
}
```

Slika 3.3 Primjer rezultat HTTP zahtjeva za registraciju u slučaju uspješne registracije

Unutar povratne informacije korisnik dobiva identifikacijski broj svog korisničkog računa te autorizacijski token koji služi za sve daljnje autorizacije i prijave.

Email adresu nije nužno navesti, no ako je navedena, korisnik će dobiti e-mail koji sadrži link prema formi za potvrdu e-mail adrese, čiji izgled je prikazan na slici 3.4.



Slika 3.4. Izgled forme za potvrdu email adrese

Prilikom svake registracije ili prijave, moguće je navesti opcionalni parametar nazvan "firebaseToken", što predstavlja token u Android aplikaciji. Taj token omogućava poslužitelju da šalje mobilne obavijesti na taj uređaj.

Prijava se može obaviti sa korisničkim imenom i lozinkom ili autentikacijskim tokenom. Prijava pomoću korisničkog imena i lozinke se obavlja pomoću HTTP zahtjeva opisanog na slici 3.5.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/userAuth/login/creds

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "username": "Kristian",
    "password": "lozinka",
    "firebaseToken": "jssdf-5346-reterterzt"
}
```

Slika 3.5. Opis HTTP zahtjeva za prijavu korisnika pomoću korisničkog imena i lozinke

U slučaju uspješne prijave dobije se odgovor identičan onome iz registracije opisan na slici 3.3., a ukoliko je prijava neuspješna dobije se HTTP odgovor sa status kodom 400.

Prijava pomoću tokena se obavlja pomoću HTTP zahtjeva opisanog na slici 3.6.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/userAuth/login/token

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "authToken": "Autentikacijski token",
    "firebaseToken": "jssdf-5346-reterterzt"
}
```

Slika 3.6. Opis HTTP zahtjeva za prijavu korisnika pomoću autentikacijskog tokena

Autentikacija unutar websocket konekcije obavlja se tijekom prve poruke nakon nastanka websocket konekcije. Ovisno o tome je li riječ o autentikaciji korisnika ili uređaja imamo dvije vrste poruke za autentikaciju. Poruke su opisane na slikama 3.7. i 3.8.

```
Tijelo poruke:

{
    "messageType": "connectUser",
    "data": {
        "authToken": "abcde_primjerTokena",
        "frontEndType": 2,
    }
}
```

Slika 3.7. Opis tijela poruke za autentikaciju korisnika u websocket komunikaciji

```
Tijelo poruke:

{
    "messageType": "connectDevice",
    "data": {
        "deviceKey": "tajni ključ identifikacije uređaja"
    }
}
```

Slika 3.8. Opis tijela poruke za autentikaciju uređaja u websocket komunikaciji

Parametar „messageType“ striktno mora biti u jednoj od dvije vrijednosti : „connectUser“ ili „connectDevice“ ovisno o dva tipa poruka.

„AuthToken“ parametar predstavlja token koji korisnik dobije prilikom prijave ili registracije. FrontendType parametar predstavlja numeracija koja određuje tip uređaja s kojeg se korisnik spaja te se nude sljedeće vrijednosti:

0: Web aplikacija

1: Web aplikacija u responzivnom načinu rada

2: Android

3: wearOS

Ovo ostavlja mogućnost provjere o kojem tipu korisnika se radi u slučaju da se implementacija proširi na wearOS i na Web te da će poslužitelj moći razlikovati između različitih tipova instanci.

„deviceKey“ parametar predstavlja tajni ključ koji identificira uređaj.

3.2.2. Upravljanje korisničkim računom

Korisničkim računom je moguće upravljati promjenom lozinke, odjavljivanjem sa svih sesija, dodavanjem e-mail adrese računu, oporavkom računa i brisanjem računa.

Odjavljivanje se ostvaruje sa HTTP pozivom opisanom na slici 3.9. Ovisno o parametru „logoutOtherSessions“ obavit će se normalno odjavljivanje ili odjavljivanje svih sesija. Odgovor na zahtjev o odjavljivanju je prazna poruka sa HTTP kodom 200 ili 400, ovisno o uspješnosti odjave.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/userAuth/logout

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "authToken": "Autentikacijski token korisnika",
    "logoutOtherSessions": true
}
```

Slika 3.9. Opis HTTP zahtjeva za odjavljivanje korisnika

Odjavljivanje sa svih sesija postiže se brisanjem autentikacijskih tokena i websocket porukom. Stoga će bilo koji uređaj koji je prijavljen na taj korisnički račun pri sljedećoj prijavi biti zaustavljen pri pokušaju prijave s tokenom. Samo brisanje tokena ne rješava cijeli problem jer i dalje postoje korisnici koji su trenutno prijavljeni i imaju aplikaciju otvorenu, iako im poslužitelj neće dozvoliti pristup nikakvim daljnjim podacima jer je za svaku komunikaciju s poslužiteljem potreban token. Ti korisnici prisilno se odjavljuju putem posebne poruke koja se šalje svim tim korisnicima putem websocket konekcije. U toj poruci se korisniku navodi i razlog odjavljivanja, koji može biti brisanje korisničkog računa, prisilno odjavljivanje sa svih sesija ili promjena lozinke. Primjer poruke je na slici 3.10. Parametar „logoutReason“ je enumeracija razloga za odjavljivanje i može poprimiti sljedeće vrijednosti:

0: Korisnički račun je obrisan

1: Lozinka je promijenjena te je korisnik zatražio odjavljivanje ostalih sesija

2: Korisnik je zatražio odjavljivanje svih sesija

3: Korisnik je zatražio odjavu trenutne sesije

Nakon što poslužitelj pošalje poruku o prisilnom odjavljivanju nakon 3 sekunde će prekinuti websocket vezu.

```
Tijelo poruke:

{
    "messageType": "userMessage",
    "data": {
        "logoutReason": 0
    }
}
```

Slika 3.10. Opis tijela poruke za prisilno odjavljivanje u websocket komunikaciji

Promjena lozinke se ostvaruje sa HTTP pozivom opisanim na slici 3.11.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/userAuth/changePassword

Tip podatka u tijelu zahtjeva: application/json


Tijelo zahtjeva:
{
    "userId": 10
    "oldPassword": "stara lozinka",
    "newPassword": "nova lozinka",
    "logoutOtherSessions": true,
    "dontLogoutToken": "Autentikacijski token korisnika"
}
```

Slika 3.11. Opis HTTP zahtjeva za promjenu lozinke korisnika

Unutar poziva je potrebno navesti staru lozinku, novu lozinku, informaciju dali se želi odjaviti ostale sesije te ako da, navesti token koji se želi ostaviti (trenutni token korisnika). Odgovor na ovaj zahtjev je HTTP status kod 200 ili 400 ovisno o uspješnosti promjene lozinke.

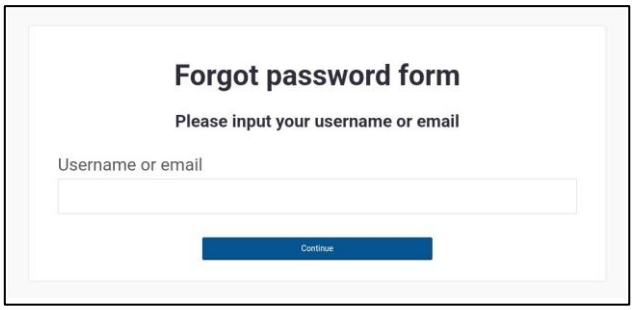
Ukoliko pri registraciji nije navedena email adresa moguće ju je dodati kasnije HTTP pozivom opisanim na slici 3.12. Odgovor na ovaj zahtjev je HTTP status kod 200 ili 400 ovisno o uspješnosti početka procesa dodavanja email adrese.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/userAuth/addEmail

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "token": "Autentikacijski token korisnika",
    "email": "kristiankliskovic@student.ferit.hr"
}
```

Slika 3.12. Opis HTTP zahtjeva za dodavanje email adrese

Nakon obavljanja HTTP poziva šalje se mail o potvrdi email adrese te je postupak potvrde isti kao i pri registraciji sa email adresom.

Ukoliko korisnik zaboravi lozinku moguć je oporavak računa ukoliko račun ima email adresu. Umjesto direktnog HTTP poziva implementirana je jednostavna forma za unos email adrese ili korisničkog imena prema slici 3.13. Relativna web adresa na kojoj se može pronaći ova forma je „/email/forgotPassword“



Slika 3.13. Izgled forme za oporavak lozinke

Ako se unese korisničko ime korisnik dobije jednu od tri poruke: „Ne postoji korisnik s tim imenom“, „Navedeni korisnik nema povezanu email adresu sa računom“ ili „Email za oporavak računa je poslan na 'email adresa' “. U zadnjoj poruci se zbog sigurnosti središnji dio email adrese zamjeni sa zvjezdicama.

Ako se unese email dobije se jedna od dvije poruke: „Email nije povezan sa nijednim korisnikom“ ili „Email za oporavak računa je poslan na 'email adresa' “.

Brisanje korisničkog računa se ostvaruje sa HTTP pozivom opisanom na slici 3.14.

```
Vrsta HTTP zahtjeva: POST
Relativna API putanja: /API/userAuth/delete
Tip podatka u tijelu zahtjeva: application/json
Tijelo zahtjeva:
{
    "authToken": "Autentikacijski token korisnika"
}
```

Slika 3.14. Opis HTTP zahtjeva za brisanje korisničkog računa

U slučaju da je korisnik administrator nekih uređaja on neće moći obrisati svoj korisnički račun jer bi to značilo brisanje i tih uređaja, što bi predstavljalo potencijalno neželjen gubitak funkcionalnosti za druge korisnike. Stoga će prvo morati obrisati svoje uređaje ručno ili prebaciti administratorska prava na druge korisnike.

3.2.3. Upravljanje uređajem

Uređaj se dodaje u sustav slanjem HTTP poziva opisanog na slici 3.15. Potrebno je navesti željeni naziv uređaja, autentikacijski token korisnika koji dodaje uređaj te se može navesti tajni ključ uređaja. Ukoliko se ne navede taj parametar poslužitelj će sam odabrati ključ. Ukoliko je naveden ključ koji već postoji u sustavu, poslužitelj će vratiti status kod 400 te uređaj neće biti dodan. Dozvoljeno je imati više uređaja s istim imenom.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/device/addDevice

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "deviceName": "Ime uređaja",
    "deviceKey": "Željeni ključ uređaja",
    "authToken": "Autentikacijski token korisnika"
}
```

Slika 3.15. Opis HTTP zahtjeva za dodavanje uređaja

Brisanje uređaja se obavlja slanjem HTTP zahtjeva opisanog na slici 3.16. Potrebno je navesti autentikacijski token koji pripada administratoru tog uređaja te identifikacijski broj tog uređaja.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/device/deleteDevice

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "deviceId": 31,
    "authToken": "Autentikacijski token admina uređaja"
}
```

Slika 3.16. Opis HTTP zahtjeva za brisanje uređaja

Uređaju je moguće promijeniti ime slanjem HTTP zahtjeva opisanog na slici 3.17. Potrebno je navesti autentikacijski token administratora uređaja, identifikacijski broj uređaja te novo ime uređaja.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/device/renameDevice

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "deviceId": 31,
    "deviceName": "Novo ime uređaja",
    "authToken": "Autentikacijski token administratora uređaja"
}
```

Slika 3.17. Opis HTTP zahtjeva za mijenjanje imena uređaja

Grupe i kompleksne grupe te polja i stanja koja sadrže mijenjaju se pomoću HTTP poziva navedenom u prilogu 1. Potrebno je navesti tajni ključ uređaja, listu grupa i kompleksnih grupa te njihovu unutarnju strukturu.

3.2.4. Upravljanje stanjem uređaja

Konačno, da bi se ostvarila glavna funkcionalnost sustava potrebno je moći upravljati stanjima unutar samih uređaja. Stanjem uređaja se može upravljati na slijedeće načine: mijenjanjem vrijednosti polja unutar grupe, mijenjanjem stanja kompleksne grupe te mijenjanjem vrijednosti u polju unutar kompleksne grupe. Različiti su pozivi ako stanje mijenja uređaj i ako stanje mijenja korisnik zbog drugačije autentikacije dva entiteta.

Ako se radi o mijenjanju stanja od strane uređaja stavlja se parametar „deviceKey“ i API putanja završava na „/device“.

Ako je riječ o korisniku stavlja se „authToken“ koji sadrži autentikacijski token korisnika i API putanja završava na „/user“ te se mora navesti parametar „deviceId“ koji specificira uređaj. Autentikacijski token osim što mora biti važeći, mora pripadati korisniku koji ima dopuštenje na uređaj, dopuštenja su detaljno objašnjena u potpoglavlju 3.2.5.

Mijenjanje stanja polja unutar grupe se postiže HTTP zahtjevom prema primjeru na slici 3.18.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/device/changeField/device

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "deviceKey": "tajni ključ identifikacije uređaja",
    "groupId": 0,
    "fieldId": 0,
    "fieldValue": 22.5
}
```

Slika 3.18. Opis HTTP zahtjeva za mijenjanje stanja polja u grupi

Mijenjanje stanja kompleksne grupe se postiže HTTP zahtjevom prema primjeru na slici 3.19.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/device/changeField/device

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "deviceKey": "tajni ključ identifikacije uređaja",
    "groupId": 0,
    "state": 0
}
```

Slika 3.19. Opis HTTP zahtjeva za mijenjanje stanja kompleksne grupe

Mijenjanje stanja polja u kompleksnoj grupi se postiže HTTP zahtjevom prema primjeru na slici 3.20.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/device/changeField/device

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "authToken": "tajni ključ identifikacije uređaja",
    "deviceId": 1,
    "groupId": 0,
    "stateId": 0,
    "fieldId": 0,
    "fieldValue": {
        "R": 0,
        "G": 255,
        "B": 255,
    }
}
```

Slika 3.20. Opis HTTP zahtjeva za mijenjanje stanja polja unutar kompleksne grupe

Vidljivo je da parametar „fieldValue“ može biti različitog tipa ovisno o vrsti polja koje se mijenja.

Ako je polje brojčanog tipa, vrijednost mora biti veća od minimalne vrijednosti polja, manja od maksimalne vrijednosti polja te odgovarati po minimalnom koraku polja. Recimo ako je polje definirano sa:

minimum = 0
maksimum = 100
korak = 0.25

Vrijednost mora biti u intervalu [0,100], ali biti formata 0.25 * k gdje je k cijeli broj.

Ako se radi o polju sa višestrukim izborom vrijednost mora biti redni broj jednog od dostupnih stanja.

Ako se radi o RGB polju kao na slici 3.20. tada vrijednost mora biti objekt koji sadrži R, G i B parametre koji su cijeli brojevi u intervalu [0, 255].

Ako se radi o tekstualnom polju vrijednost mora biti string, te ako se radi o boolean polju vrijednost mora biti true ili false.

Kada se dogodi promjena na stanju uređaja pošalje se websocket poruka tom uređaju i svim korisnicima koji imaju prava na polje koje je promijenjeno. Uređaju se pošalje poruka koja sadrži kompletno stanje uređaja, primjer se nalazi u prilogu 2. Korisnicima trenutno spojenima na websocket server (oni koji imaju upaljenu aplikaciju) se pošalju sva stanja uređaja na koje imaju pravo (samo dijelovi tih uređaja na koje imaju pravo), primjer se nalazi u prilogu 3.

3.2.5. Upravljanje dopuštenjima na uređaj

Do sada je opisivana komunikacija između uređaja i poslužitelja te između administratora uređaja i poslužitelja. U osnovi, samo administrator uređaja ima pravo komunikacije sa uređajem (može vidjeti i mijenjati stanja), no može podijeliti dopuštenja ostalim korisnicima. Dopuštenja mogu biti na cijeli uređaj, grupu, polje unutar grupe i na kompleksnu grupu te može omogućavati korisnicima samo pregled vrijednosti polja ili mijenjanje vrijednosti polja. Dopuštenja su hijerarhijski ustrojena, odnosno dodavanjem dopuštenja čitanja na grupu brišu se sva dopuštenja čitanja na pojedinačna polja unutar te grupe jer se podrazumijevaju. Dodavanjem dopuštenja čitanja na cijeli uređaj brišu se dopuštenja čitanja na sve entitete unutar uređaja te nova naredba preuzima funkciju. Korisnik može imati dopuštenje čitanja na cijeli uređaj i dopuštenje čitanja i pisanja na pojedina polja ili grupe u raznim kombinacijama. Kada korisnik ima dopuštenje čitanja i pisanja na cijeli uređaj ne može imati nikakva daljnja dopuštenja jer se podrazumijevaju, pri takvom stanju on ima jednaka prava kao i administrator uređaja, osim prava da uređaj obriše, promjeni admina i promjeni naziv uređaja.

Dopuštenja se mogu dodati preko nekoliko HTTP poziva prema primjerima na slikama 3.21, 3.22, 3.23 i 3.24.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/userRights/addDeviceRight

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "authToken": "Autentikacijski token admina uređaja",
    "userId": 3,
    "deviceId": 1,
    "readOnly": true
}
```

Slika 3.21. Opis HTTP zahtjeva za dodavanja prava čitanja na cijeli uređaj

Za dodavanja prava na cijeli uređaj potrebno je navesti autentikacijski token koji pripada administratoru uređaja, identifikacijski broj korisnika kojem se želi dati dopuštenje, identifikacijski broj uređaja te informaciju radi li se o pravu čitanja ili pisanja.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/userRights/addGroupRight

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "authToken": "Autentikacijski token admina uređaja",
    "userId": 3,
    "deviceId": 1,
    "groupId": 1,
    "readOnly": true
}
```

Slika 3.22. Opis HTTP zahtjeva za dodavanja prava čitanja i pisanja na grupu

Pri dodavanju dopuštenja na grupu potrebno je osim svih podatka kao i za dodavanje dopuštenja na uređaj dodati i identifikacijski broj grupe.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/userRights/addFieldRight

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "authToken": "Autentikacijski token admina uređaja",
    "userId": 3,
    "deviceId": 1,
    "groupId": 1,
    "fieldId": 1,
    "readOnly": true
}
```

Slika 3.23. Opis HTTP zahtjeva za dodavanja prava čitanja i pisanja na polje

Pri dodavanju dopuštenja na polje unutar grupe potrebno je osim svih podatka kao i za dodavanje dopuštenja na grupu dodati i identifikacijski broj polja.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/userRights/addComplexGroupRight

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "authToken": "Autentikacijski token admina uređaja",
    "userId": 3,
    "deviceId": 1,
    "complexGroupId": 1,
    "readOnly": true
}
```

Slika 3.24. Opis HTTP zahtjeva za dodavanja prava čitanja i pisanja na kompleksnu grupu

Pri dodavanju dopuštenja na kompleksnu grupu potrebno je osim svih podatka kao i za dodavanje dopuštenja na uređaj dodati i identifikacijski broj kompleksne grupe.

Administrator može i pregledati sva dopuštenja na uređaj pomoću HTTP zahtjeva opisnog na slici 3.25. i Prilogu 4. Potrebno je poslati autentikacijski token koji pripada administratoru tog uređaja te identifikacijski broj uređaja čijim se dopuštenjima želi pristupiti.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/userRights/getUserPermissions

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "authToken": "Autentikacijski token admina uređaja",
    "deviceId": 1
}
```

Slika 3.25. Opis HTTP zahtjeva za dobivanje svih prava na uređaj

Brisanje dopuštenja je moguće napraviti pomoću HTTP zahtjeva opisanog na slici 3.26. Zahtjev mora sadržavati autentikacijski token koji pripada administratoru uređaja, identifikacijski broj korisnika čije se dopuštenje želi obrisati te identifikacijski broj uređaja. Različiti su zahtjevi za brisanje dopuštenja na uređaj, grupu, polje u grupi i kompleksnu grupu. API putanja može završavati na „/deleteDeviceRight“, „/deleteGroupRight“, „/deleteFieldRight“ ili „/deleteComplexGroupRight“, te sadržavati neke ili sve od parametara „groupId“, „fieldId“ i „complexGroupId“.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/userRights/deleteFieldRight

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "authToken": "Autentikacijski token admina uređaja",
    "deviceId": 1,
    "userId": 5,
    "groupId": 2,
    "fieldId": 7
}
```

Slika 3.26. Opis HTTP zahtjeva za brisanje dopuštenja na polje u grupi

Pri dodavanju ili brisanju prava dolazi do slanja websocket poruke korisnicima, ta poruka sadrži informacije o stanju svih uređaja na koje ima pravo.

3.2.6. Upravljanje okidačima

Ova platforma odnosno njezin poslužitelj omogućava postavljanje okidača koji mogu promatrati stanje uređaja te na nekakav uvjet ispuniti neku radnju. U ovom potpoglavlju ću opisati sve podatke koje okidač mora sadržavati, te kako se dodaju i uklanjaju.

Okidači imaju inicijaciju i reakciju. Inicijacija okidača može biti vrijednosna (polje u grupi ili polje u kompleksnoj grupi je postiglo postavljenu vrijednost) ili vremenska. To su ukupno 3 vrste inicijacije jer se okidač vezan za polje u grupi i polje u kompleksnoj grupi smatraju odvojenima. Vrste inicijacije mogu se predstaviti enum klasom prema slici 3.27.

```
export enum ETriggerSourceType {
    FieldInGroup,
    FieldInComplexGroup,
    TimeTrigger,
}
```

Slika 3.27. Enum klasa za vrstu inicijacije okidača

Ako se radi o vrijednosnoj inicijaciji potrebno je navesti adresu polja te vrijednost za koju se okidač okida. Adresa polja je skup podataka koji definiraju o kojem polju se radi. Sadrži identifikacijski broj uređaja, grupe i polja ako se radi o polju u grupi ili identifikacijski broj uređaja, kompleksne grupe, stanja i polja ako se radi o polju u kompleksnoj grupi. Ove dvije vrste adresa se prikazuju sučeljima na slikama 3.28. i 3.29.

```
export interface ITrigSourceFG {
    deviceId: number,
    groupId: number,
    fieldId: number,
}
```

Slika 3.28. Sučelje za adresu inicijacijskog polja u grupi

```
export interface ITrigSourceFCG {
    deviceId: number,
    complexGroupId: number,
    stateId: number,
    fieldId: number,
}
```

Slika 3.29. Sučelje za adresu inicijacijskog polja u grupi

Ovisno o vrsti polja moguće je imati nekoliko vrsta postavki inicijacijske vrijednosti okidača.

Ako se radi o brojčanom polju korisnik bira između 5 vrsta načina okidanja okidača. Može odabrati jednu od mogućih vrijednosti brojčanog polja te postaviti da se okidač okida kada vrijednost bude veća, manja ili jednaka toj vrijednosti ili odabrati dvije vrijednosti te postaviti da se okidač okida

kada je vrijednost polja unutar tog intervala ili izvan tog intervala. Vrste okidanja brojčano iniciranog okidača se mogu predstaviti enum klasom prema slici 3.30.

```
export enum ENumericTriggerType {
    Bigger,
    Smaller,
    Equal,
    Between,
    NotBetween,
}
```

Slika 3.30. Enum klasa za vrste okidanja numerički iniciranog okidača

Postavke okidanja numerički iniciranog okidača se sastoje od jedne ili dvije vrijednosti koje predstavljaju vrijednost ili interval okidanja te vrstu okidanja i može predstaviti sučeljem prema slici 3.31.

```
export interface INumTrig {
    value: number,
    second_value: number | null,
    type: ENumericTriggerType,
}
```

Slika 3.31. Sučelje za inicijacijske postavke numerički iniciranog okidača

Ako se radi o tekstualnom polju onda je potrebna tekstualna vrijednost te podatak hoće li se okidač okinuti ako vrijednost počinje, završava, sadrži, jest jednaka ili nije jednaka tom stringu. Vrste okidanja tekstualno iniciranog okidača mogu se prikazati enum klasom prema slici 3.32.

```
export enum ETextTriggerType {
    StartsWith,
    EndsWith,
    Contains,
    IsEqualTo,
    IsNotEqualTo,
}
```

Slika 3.32. Enum klasa za vrste okidanja tekstualno iniciranog okidača

Postavke okidanja tekstualno iniciranog okidača se sastoje od tekstualne vrijednosti te vrste okidanja i može predstaviti sučeljem prema slici 3.33.

```
export interface ITextTrig {
    value: string,
    type: ETextTriggerType,
}
```

Slika 3.33. Sučelje za inicijacijske postavke tekstualno iniciranog okidača

Ako se radi o boolean polju inicijacijske postavke su jednostavnije, sadrže samo vrijednost okidanja „true“ ili „false“, kao prema primjeru 3.34. To sučelje sadrži samo parametar „value“ koji je tipa „boolean“.

Ako se radi o polju sa višestrukim izborom potrebno je imati podatke o željenoj vrijednosti (koje predstavlja traženo stanje polja) te informaciju treba li okidač okinuti kada je vrijednost jednaka toj vrijednosti ili kad nije jednaka toj vrijednosti. Vrste okidanja polja sa višestrukim izborom možemo prikazati enum klasom prema slici 3.34., a postavke okidanja polja možemo prikazati sučeljem prema slici 3.35.

```
export enum EMCTriggerType {
    IsEqualTo,
    IsNotEqualTo,
}
```

Slika 3.34. Enum klasa za vrste okidanja okidača poljem sa višestrukim izborom

```
export interface IMCTrig {
    value: number,
    type: EMCTriggerType,
}
```

Slika 3.35. Sučelje za inicijacijske postavke okidača iniciranog polje sa višestrukim izborom

Naposljetku, ako se radi o RGB polju postavke su identične kao i za numeričko polje, samo što je potrebno dodati RGB kontekst, odnosno informaciju promatra li se R, G ili B vrijednost unutar polja. Sučelje za inicijacijske postavke okidača na RGB polju je prikazano na slici 3.36. Parametar „type“ je enum klasa sa vrijednostima „R“, „G“ i „B“.

```
export interface IRGBTrig {
    value: number,
    second_value: number | null,
    contextType: ERGBTriggerType_context,
    type: ERGBTriggerType_numeric,
}
```

Slika 3.36. Sučelje za inicijacijske postavke numerički iniciranog okidača

Ako je riječ o vremenski iniciranom okidaču potrebno je navesti vrstu ponavljanja vremenskog okidača (jedanput, dnevno i tjedno) te datum i vrijeme prvog (ili jedinog) okidanja u obliku ISO stringa. Izbor između tipa okidanja se predstavlja enum klasom prikazanom na slici 3.37. Postoji ograničenje na odabir vremena, a to je da se okidanje može postaviti u intervalima od 5 minuta, odnosno vrijeme okidanja može biti 8:00 ili 8:05, ali ništa između. Postavke vremenskog okidača prikazane su sučeljem na slici 3.38.

```
export enum ETriggerTimeType {
    Once,
    Daily,
    Weekly,
}
```

Slika 3.37. Enum klasa za vrste okidanja vremenskog okidača

```
export interface ITrigSourceTime {
    type: ETriggerTimeType,
    firstTimeStamp: string,
}
```

Slika 3.38. Sučelje za inicijacijske postavke vremenskog okidača

S ovim su objašnjene sve vrste okidanja okidača i njihove postavke.

Postoji 3 vrste reakcija na okidač, postavljanje vrijednosti nekog polja, slanje elektroničke pošte ili slanje mobilne notifikacije tom korisniku. Takav izbor okidanja se predstavlja enum klasom prema slici 3.39. koja sadrži 4 opcije gdje se odvojeno prikazuje odabir postavljanja vrijednosti unutar polja u grupi i polja u kompleksnoj grupi.

```
export enum ETrigRespType {
    Email,
    MobileNotification,
    SettingValue_fieldInGroup,
    SettingValue_fieldInComplexGroup,
}
```

Slika 3.39. Enum klasa za vrste reakcije na okidač

Nadalje za svaki od ta 4 okidanja postoje postavke reakcije. Ako je riječ o elektroničkoj pošti potrebno je navesti naslov mail-a i tekst mail-a. Postavke email reakcije mogu se prikazati sučeljem prema slici 3.40.

```
export interface ITrigRespEmail {
    emailSubject: string,
    emailText: string,
}
```

Slika 3.40. Sučelje za reakcije postavke za elektroničku poštu

Ako je riječ o mobilnoj notifikaciji potrebno je postaviti naslov notifikacije te sadržaj notifikacije, takve postavke mogu se prikazati sučeljem prema slici 3.41.

```
export interface ITrigRespMobNot {
    notificationTitle: string,
    notificationText: string,
}
```

Slika 3.41. Sučelje za reakcije postavke za mobilne notifikacije

Ukoliko je riječ o postavljanju vrijednosti polja unutar grupe potrebno je navesti adresu polja (identifikacijski broj uređaja, grupe i polja) te vrijednost na koje se to polje želi postaviti, te ukoliko je riječ o RGB polju potrebno je navesti i RGB kontekst odnosno koju od 3 vrijednosti se želi postaviti. Takve postavke se prikazuju sučeljem na slici 3.42. Parametar „value“ može biti tipa „number“ ili „string“ stoga je radi manjeg broja sučelja prikazan tipom „any“.

```
export interface ITrigSourceFG {
    deviceId: number,
    groupId: number,
    fieldId: number,
    value: any,
    rgbContext: ERGBTriggerType_context,
}
```

Slika 3.42. Sučelje za postavke reakcije postavljanje polja u grupi

Ukoliko je riječ o postavljanje vrijednosti polja unutar kompleksne grupe onda se primjenjuje sučelje prema slici 3.43.

```
export interface ITrigRespFCG {
    deviceId: number,
    complexGroupId: number,
    complexGroupStateId: number,
    fieldId: number,
    value: any,
    rgbContext: ERGBTriggerType_context,
}
```

Slika 3.43. Sučelje za postavke reakcije postavljanje polja u kompleksnoj grupi

Sada su svojstva okidača u potpunosti opisana. Okidač se može prikazati sučeljem prema slici 3.44. Vidimo da većina parametara poprima drugačiji tip ovisno o vrsti inicijacije ili reakcije.

Parametar „id“ koji predstavlja identifikacijski broj okidača, „name“ predstavlja naziv okidača, „userId“ koji predstavlja identifikacijski broj korisnika koji je postavio okidač te „fieldType“ koji označava vrstu inicijacijskog polja čija vrijednost može biti „numeric“, „text“, „boolean“, „multipleChoice“ ili „RGB“.

```
export interface ITrigger {
    id: number,
    name: string,
    userId: number,
    sourceType: ETrigSourceType
    sourceData: ITrigSourceFG | ITrigSourceFCG | ITrigSourceTime,
    fieldType: string,
    settings: INumTrig | ITextTrig | IBoolTrig | IMCTrig | IRGBTrig,
    responseType: ETrigRespType,
    responseSettings: ITrigRespEmail | ITrigRespMobNot | ITrigRespFG |
ITrigRespFCG,
}
```

Slika 3.44. Sučelje za postavke reakcije postavljanje polja u grupi

Okidač se dodaje pomoću HTTP zahtjeva opisanog na slici 3.45. Potrebno je navesti autentikacijski token korisnika koji želi dodati okidač u bazu te sve informacije navedene u sučelju „ITrigger“ izuzev parametara „id“ i „userId“ koje će server sam popuniti te spremiti podatke u bazu nakon što provjeri validnost okidača.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/triggers/addTrigger

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "authToken": "Autentikacijski token korisnika",
    "trigger": ITrigger
}
```

Slika 3.45. Opis HTTP zahtjeva za dodavanja prava čitanja i pisanja na grupu

Korisnik može pregledati sve okidače koje je unio u bazu pomoću HTTP poziva opisanog na slici 3.46. Potrebno je poslati autentikacijski token korisnika. Ako je dohvaćanje okidača iz baze podataka bilo neuspješno, server će vratiti HTTP status 400, a ako je uspješno vratiti će listu okidača prema slici 3.47.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/triggers/getAllTriggers

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "authToken": "Autentikacijski token korisnika"
}
```

Slika 3.46.Opis HTTP zahtjeva za pregled svih okidača

```
HTTP status kod: 200 OK

Tijelo odgovora:
{
    "triggers": ITrigger[]
}
```

Slika 3.47. Primjer rezultata HTTP zahtjeva za pregled svih okidača

Okidači se brišu pomoću HTTP poziva prikazanog na slici 3.48. Potrebno je navesti autentikacijski token vlasnika okidača te identifikacijski broj okidača. Ovisno o uspješnosti brisanja okidača vraća se HTTP status kod 200 ili 400.

```
Vrsta HTTP zahtjeva: POST

Relativna API putanja: /API/triggers/deleteTrigger

Tip podatka u tijelu zahtjeva: application/json

Tijelo zahtjeva:
{
    "authToken": "Autentikacijski token admina uređaja",
    "triggerId": 0,
}
```

Slika 3.48. Opis HTTP zahtjeva za brisanje okidača

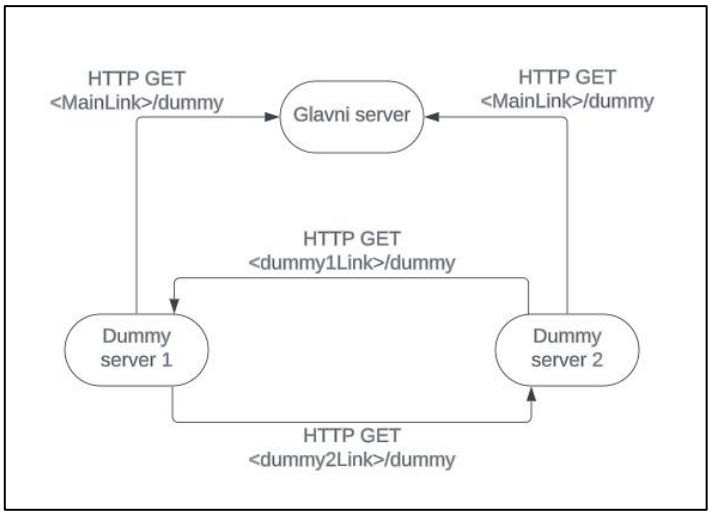
3.2.7. Hostanje servera

Hostanje servera je moguće izvesti na velikom broju komercijalno dostupnih hostinga, jedni od najpopularnijih među njima su Heroku hosting [5] i Render hosting [6]. Render i Heroku nude približno istu uslugu, sa glavnim razlikama u cijeni najmanjeg paketa koji je u oba slučaja dovoljan za projekte ovog tipa. U tablici 3.1 navesti ću bitne specifikacije oba Hostinga čiji su izvori dokumentacije oba Hostinga, osobno iskustvo i mojih istraživanja tuđih iskustava.

Tablica 3.1. Usporedba Heroku i Render hostinga

	Render hosting	Heroku hosting
Maksimalna alocirana radna memorija	512 MB	512 MB
Minimalni izmjerni RTT za WS poruku	Ne zamjetno mali	Ne zamjetno mali
Mjesečna cijena	0€	7€
Vrijeme neaktivnosti nakon koje se server ugasi	15 min	Neograničeno
Maksimalno trajanje WS konekcije	300 sekundi	Neograničeno
Broj besplatnih sati mjesečno	750 sati (31.25 dana)	-
Automatsko depojanje sa github-a	Da	Da

Vidljivo je da Heroku ima znatno bolje parametre. Maksimalno trajanje web socket konekcije od 300 sekundi na Render platformi bi potencijalno stvaralo probleme u radu sustava, no implementirano je i na mikrokontrolerskoj strani i android strani da čim web socket konekcija pukne, odmah se pokušava stvoriti nova stoga to nije velik problem. Problem koji je malo teže rješiv je gašenje servera nakon perioda neaktivnosti od 15 minuta. Taj problem bi se mogao zanemariti ako možemo pretpostaviti da će u svakom trenutku barem jedan uređaj biti spojen na server, to nije toliko loša pretpostavka pogotovo ako uzmemo u obzir da bilo koji HTTP poziv ponovo budi server, no serveru ipak treba oko 5 minuta da se u potpunosti pokrene. Ipak, moguće je zaobići taj problem sa nekoliko „dummy“ server instanci čija bi jedina zadaća bila da međusobno „bockaju“ jedni druge sa HTTP pozivima svakih nekoliko minuta i glavni server kako nijedan od njih ne bi došao u stanje neaktivnosti. Ako se to želi implementirati pojavi se novi problem, a to je da ograničenje Rendera na 750 sati mjesečno vrijedi ukupno za sve instance što znači dok jedan server može raditi punih 31 dan, dva ili više servera bi među sobom morala dijeliti tih 750 sati. Ako se želi izbjeći to ograničenje potrebno je napraviti novi Render korisnički račun te sa njega pokrenuti novu instancu „dummy“ servera. Render zahtjeva da jedan github račun može biti povezan samo na jedan render račun stoga je potrebno napraviti i novi github korisnički račun i forkati projekt. Za ovaj projekt, odabrao sam Render hosting radi dobrih performansi i mogućnosti besplatnog plana stoga sam morao implementirati dva „dummy“ servera koja svakih 5 minuta posjećuju jedan drugoga te glavni server na relativnu putanju „/dummy“. Prikaz toka HTTP zahtjeva prikazan je na slici 3.49. Taj server također mogu iskoristiti da držim veći broj servera „budnim“.



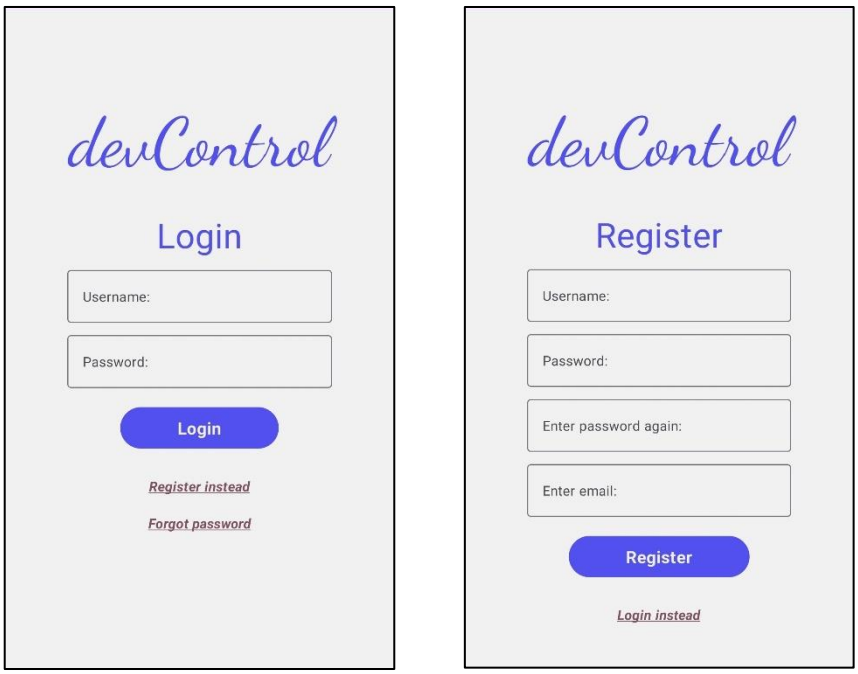
Slika 3.49. Prikaz toka HTTP zahtjeva između servera

3.3. Android aplikacija

U ovom potpoglavlju je opisano korištenje android aplikacije. Aplikacija je razvijena u Android studiju [7] koristeći Kotlin programski jezik i Jetpack compose alat za izradu korisničkog sučelja te MVVM arhitekturu.

3.3.1. Prijava i registracija

Prvi ekrani koje korisnik vidi su ekrani za prijavu i registraciju. Ti ekrani su prikazani na slici 3.50. Ekran za prijavu također nudi otvaranje forme za oporavak računa opisan u potpoglavlju 3.3.2.



Slika 3.50. a) Izgled ekrana za prijavu b) Izgled ekrana za registraciju

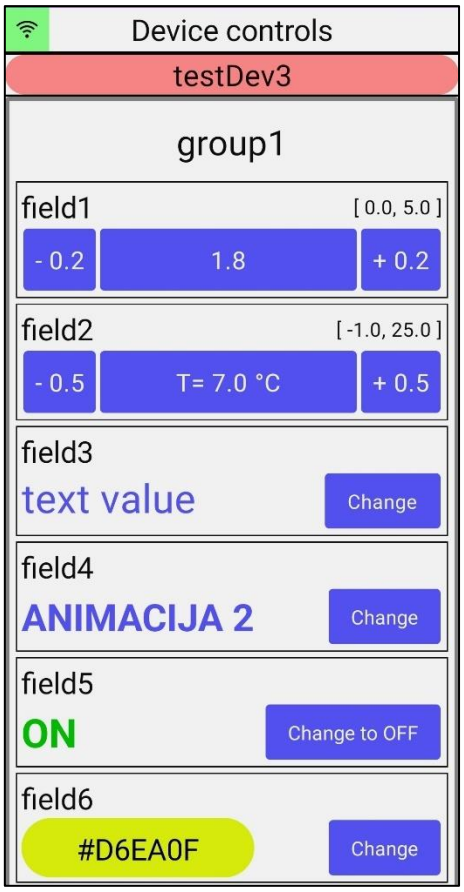
Prijavom aplikacija dobiva autentikacijski token te ga sprema u memoriju telefona. Pri idućim otvaranjima aplikacija će pokušati iskoristiti taj token za prijavu kako korisnik ne bi morao ponovno upisivati lozinku. Ako pri idućim pokretanjima aplikacije prijava pomoću tokena ne uspije korisnik će morati ponovno upisati lozinku. Nakon uspješne prijave ili registracije aplikacija se navigira na glavni ekran aplikacije.

3.3.2. Upravljanje podacima uređaja

Ekran za pregled uređaja je glavni ekran aplikacije i na njemu korisnik vidi sve uređaje na koje ima pravo. Na glavnom ekranu se još vide kontrole za navigaciju na postavke (gore desno) te status websocket konekcije na poslužitelj (gore lijevo). Ako je simbol zelene boje konekcija je aktivna te korisnik može biti siguran da vidi sve najnovije informacije vezane za svoje uređaje. Izgled ekrana prikazan je na slici 3.51.

Slika 3.51. Izgled glavnog ekrana aplikacije

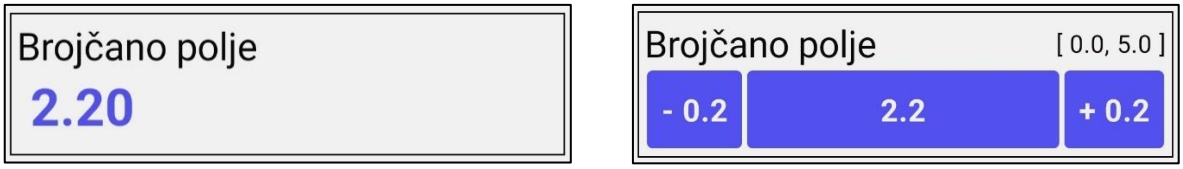
Nakon što korisnik klikne na jedan od raspoloživih uređaja aplikacija se navigira na ekran za upravljanje uređajem. Na tom ekranu korisnik vidi sve kontrole tog uređaja za koje ima dopuštenje. Izgled tog ekrana je prikazan na slici 3.52.



Slika 3.52. Izgled ekrana za kontrolu uređaja

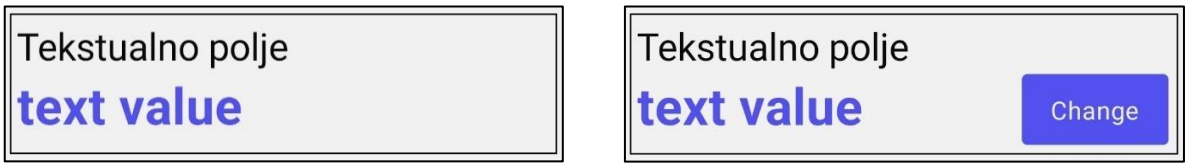
Svako od 5 polja može biti u jednosmjernom i dvosmjernom načinu rada. Jednosmjerni način rada označuje da kontrola služi samo kao prikaz podataka bez mogućnosti utjecaja na vrijednost. Dvosmjerni prikaz podataka se koristi za prikaz podataka i mijenjanje podataka.

Ako je brojčano polje u dvosmjernom načinu rada onda omogućava korisniku izmjenu vrijednosti na dva načina: precizno pomoću dvije tipke koje omogućavaju promjenu vrijednosti za iznos minimalnog koraka tog polja te grubo pomoću skočnog dijaloga koji sadrži klizač te omogućuje postavljanje vrijednosti na bilo koju validnu vrijednost za to polje. Izgledi različitih polja i načini upravljanja su prikazani na slikama od 3.53. do 3.61.

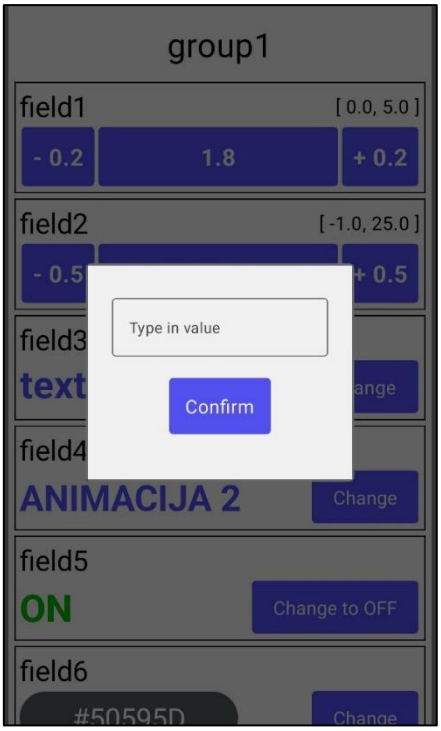


Slika 3.53 a) Izgled brojčanog polja u jednosmjernom i b) dvosmjernom načinu rada

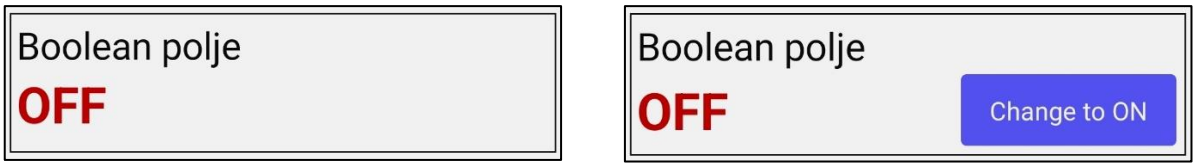
Slika 3.54. Izgled kontrole za promjenu vrijednosti brojčanog polja



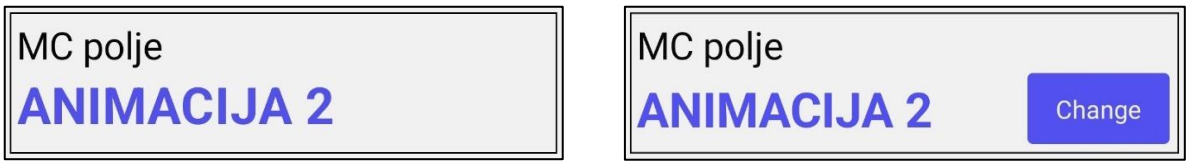
Slika 3.55 a) Izgled tekstualnog polja u jednosmjernom i b) dvosmjernom načinu rada



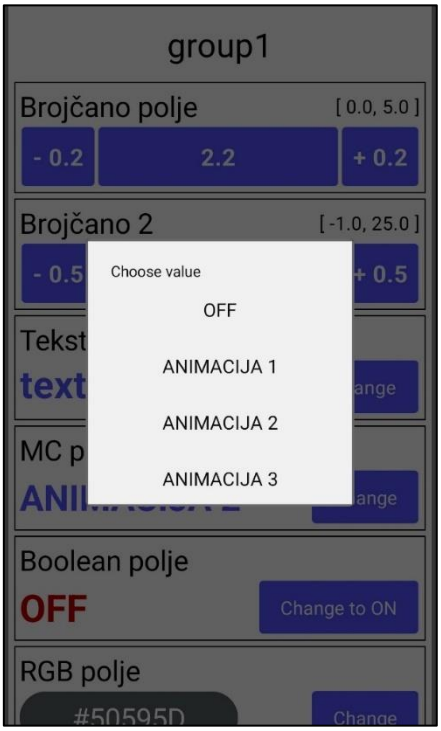
Slika 3.56. Izgled kontrole za promjenu vrijednosti tekstualnog polja



Slika 3.57 a) Izgled boolean polja u jednosmjernom i b) dvosmjernom načinu rada

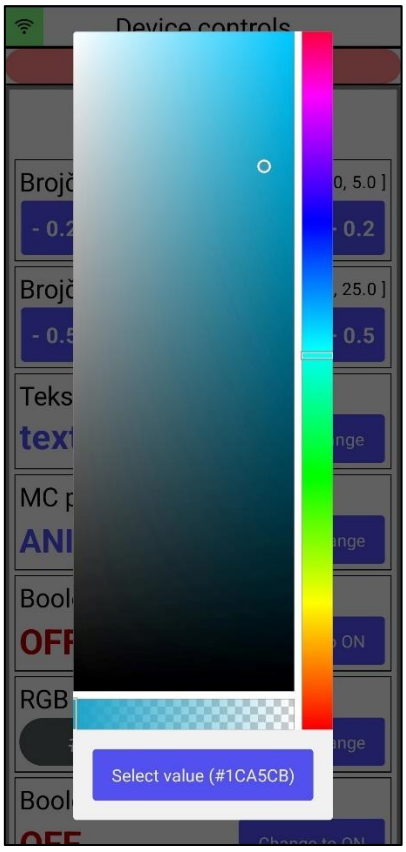


Slika 3.58. a) Izgled polja sa višestrukim izborom u jednosmjernom i b) dvosmjernom načinu rada



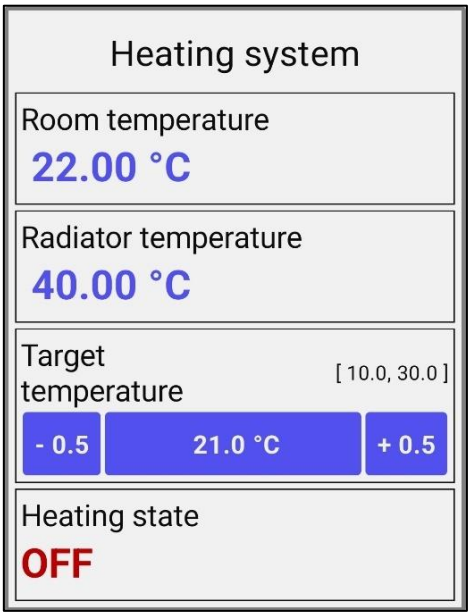
Slika 3.59. Izgled kontrole za promjenu vrijednosti polja sa višestrukim izborom

Slika 3.60. a) Izgled RGB polja u jednosmjernom i b) dvosmjernom načinu rada



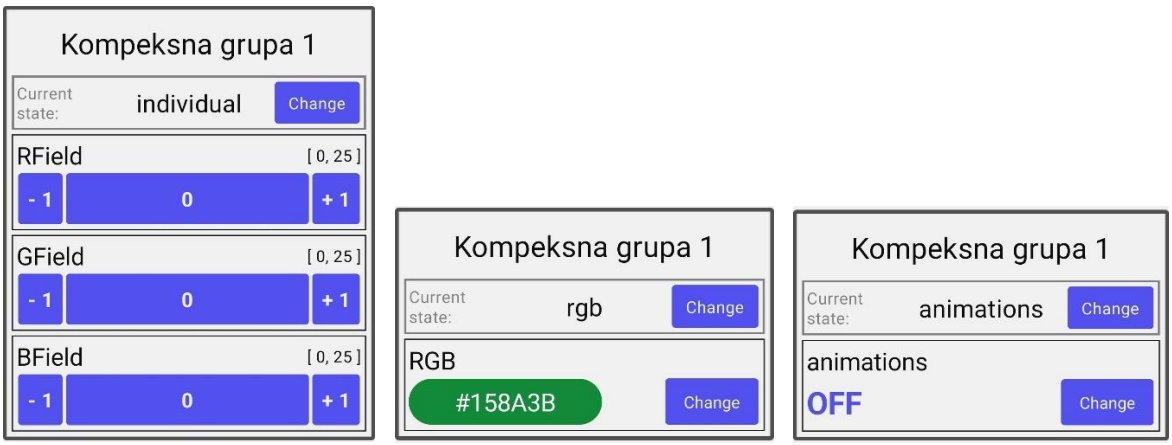
Slika 3.61. Izgled kontrole za promjenu vrijednosti RGB polja

Grupa se prikazuje sa imenom grupe te sa popisom svih polja unutar te grupe. Prikaz grupe je na slici 3.62.



Slika 3.62. Izgled grupe

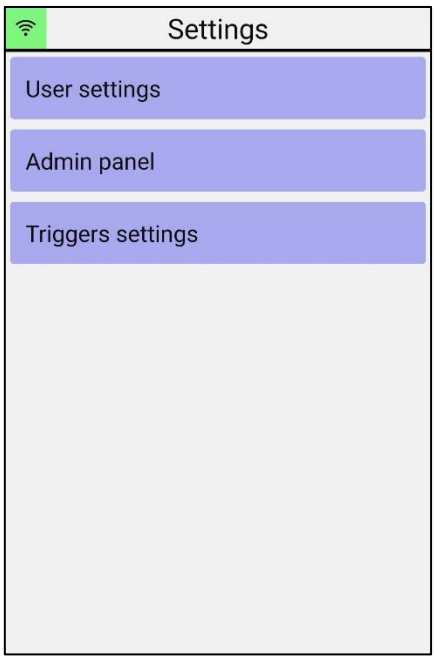
Kompleksna grupa se prikazuje sa imenom grupe, imenom trenutnog stanja, tipke za otvaranje dijaloga za promjenu stanja te popisom svih polja unutar trenutno odabranog stanja. Prikazi kompleksne grupe su na slikama 3.63.



Slika 3.63. Izgledi kompleksne grupe za kontrolu RGB trake
a) Odabir pojedinačnih svjetlina b) Odabir hex koda boje c) Odabir animacije

3.3.3. Administracija uređaja

Korisnik pritiskom tipke gore desno na glavnom ekranu aplikacije navigira se na ekran za postavke te sa sa njega može navigirati na tri ekrana: admin panel, korisničke postavke te postavke okidača kao na slici 3.64.

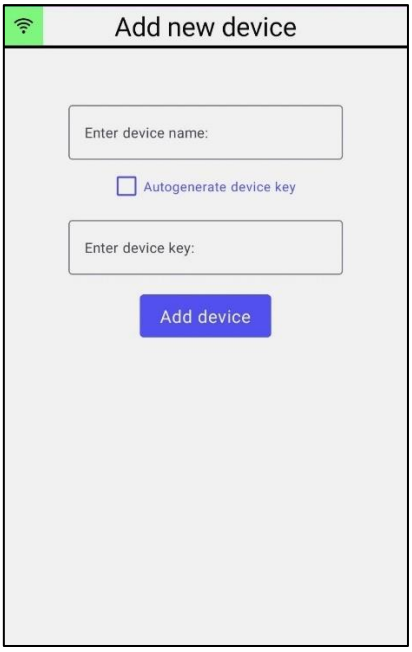


Slika 3.64. Izgled ekrana za postavke aplikacije i korisničkog računa

Otvaranjem admin panela korisniku se pokazuje popis uređaja kojima je on admin. Sa ovog ekrana korisnik se može navigirati na ekran za dodavanje uređaja. Izgled admin panela je prikazan na slici 3.65.

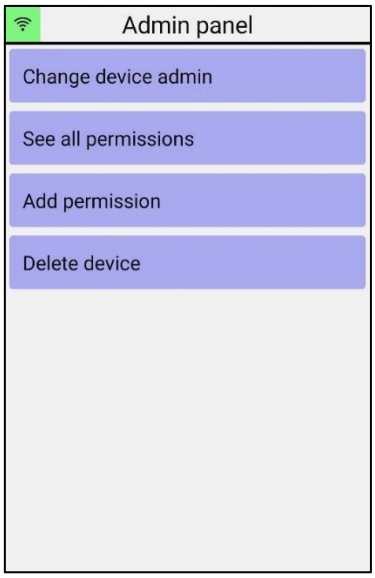
Slika 3.65.Izgled admin panela

Ekran za dodavanje uređaja služi za dodavanje novog uređaja. Da bi se dodao uređaj potrebno je navesti ime uređaja te upisati željeni tajni ključ uređaja ili odabrati automatsko generiranje ključa. Ekran za dodavanje uređaja je prikazan na slici 3.66.



Slika 3.66. Izgled ekrana za dodavanje novog uređaja

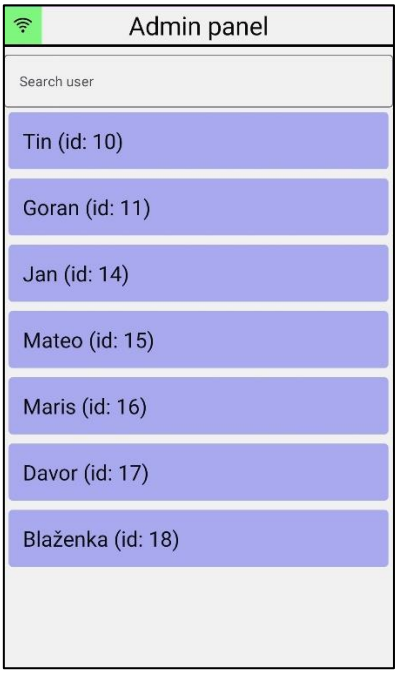
Klikom na pojedini uređaj otvaraju se administracijske postavke tog uređaja. Prikaz ekrana sa administracijskim postavkama uređaja je na slici 3.67.



Slika 3.67. Izgled ekrana za administracijske postavke uređaja

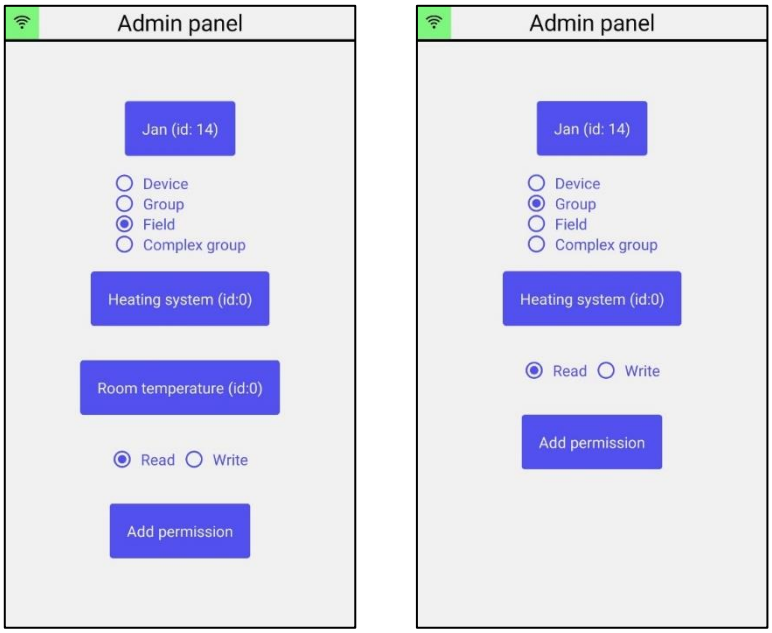
Sa ovog ekrana korisnik se može navigirati na ekran za promjenu administratora uređaja, ekran za pregled svih dopuštenja na uređaj i ekran za dodavanje dopuštenja te obrisati cijeli uređaj.

Ekran za promjenu administratora uređaja omogućava adminu da izabere drugog korisnika koji će postati administratorom uređaja. Izgled ekrana je na slici 3.68.



Slika 3.68. Izgled ekrana za promjenu administratora uređaja

Ekran za dodavanje dopuštenja služi da administrator može dodati dopuštenja na uređaj. Prvo mora odabrati kojem korisniku želi dodati dopuštenje. Potom bira na koji entitet daje dopuštenje, polje, grupu, kompleksnu grupu ili cijeli uređaj. Posljednje bira hoće li to dopuštenje biti jednosmjerno ili dvosmjerno, odnosno hoće li korisnik moći samo čitati podatke ili i mijenjati podatke. Odabir korisnika, grupe, polja i kompleksne grupe se postiže putem skočnog dijaloškog okvira. Izgled ekrana za dodavanje dopuštenja se nalazi na slici 3.69.



Slika 3.69. Izgledi ekrana za dodavanje dopuštenja

a) Dodavanje dopuštenja na polje unutar grupe b) Dodavanje dopuštenja na grupu

Ekran za pregled svih dopuštenja na uređaj služi da administrator može vidjeti sva dopuštenja na uređaj počevši od dopuštenja na cijeli uređaj, pojedinačne grupe, polja unutar grupe te kompleksne grupe. Sa tog ekrana administrator može i obrisati pojedinačna dopuštenja. Ekran za prikaz svih dopuštenja je prikazan na slici u Prilogu 5.

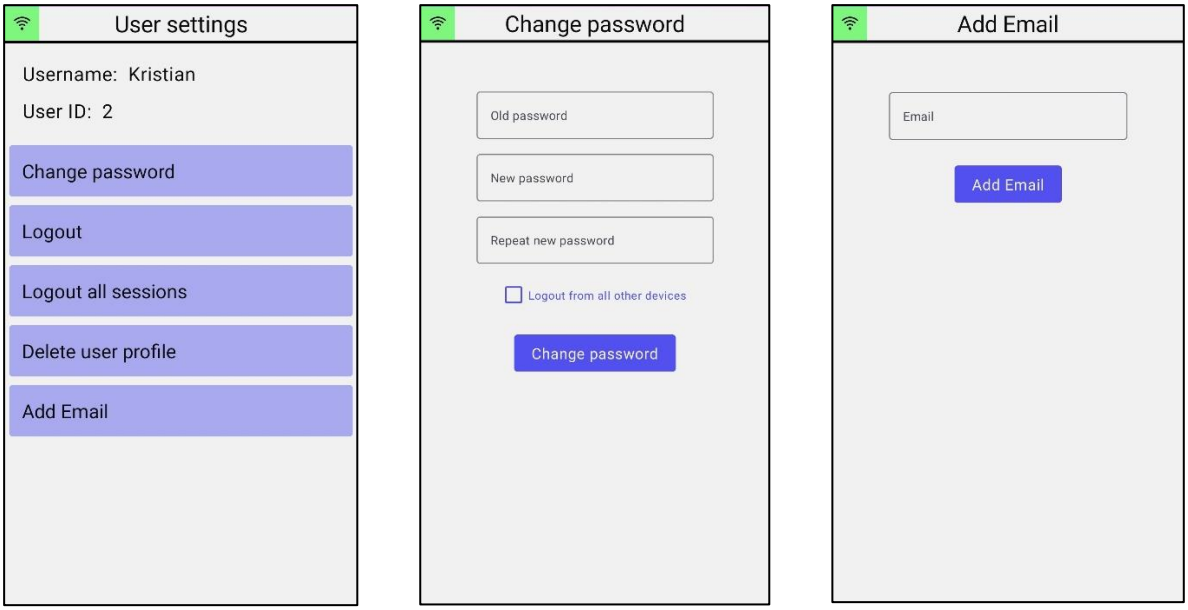
3.3.4. Upravljanje korisničkim računom

Pritiskom na tipku „User settings“ na glavnog ekranu postavki otvaraju se postavke korisničkog računa. Moguće akcije na korisničkom računu su:

1. Odjavljivanje
2. Odjavljivanje sa svih instanci tog korisnika
3. Navigiranje na ekran za promjenu lozinke – za uspješnu promjenu lozinke potrebno je upisati i trenutnu lozinku uz duplu novu lozinku, Moguće je odabrati opciju za odjavljivanje svih ostalih instanci tog korisničkog računa.

4. Brisanje korisničkog računa (brisanje nije moguće ukoliko je korisnik administrator barem jednog uređaja – mora prebaciti vlasništvo na drugog korisnika ili obrisati uređaj)
5. Dodavanje email-a. Ako korisnik nema email onda mu se prikazuje tipka za dodavanje email adrese.

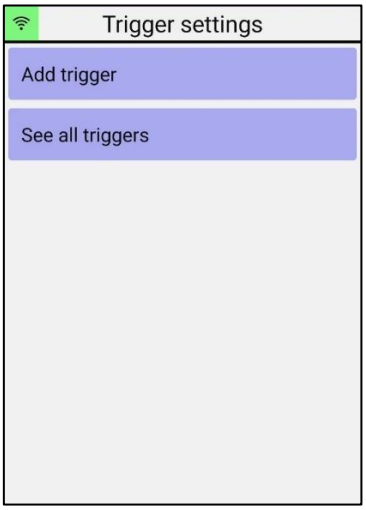
Izgledi ekrana za upravljanje korisničkim računom, ekrana za mijenjanje lozinke i ekrana za dodavanja email adrese su prikazani na slici 3.70.



Slika 3.70. a) Izgledi ekrana korisničkih postavki b) ekrana za promjenu lozinke c) ekrana za dodavanje email-a

3.3.5. Upravljanje okidačima

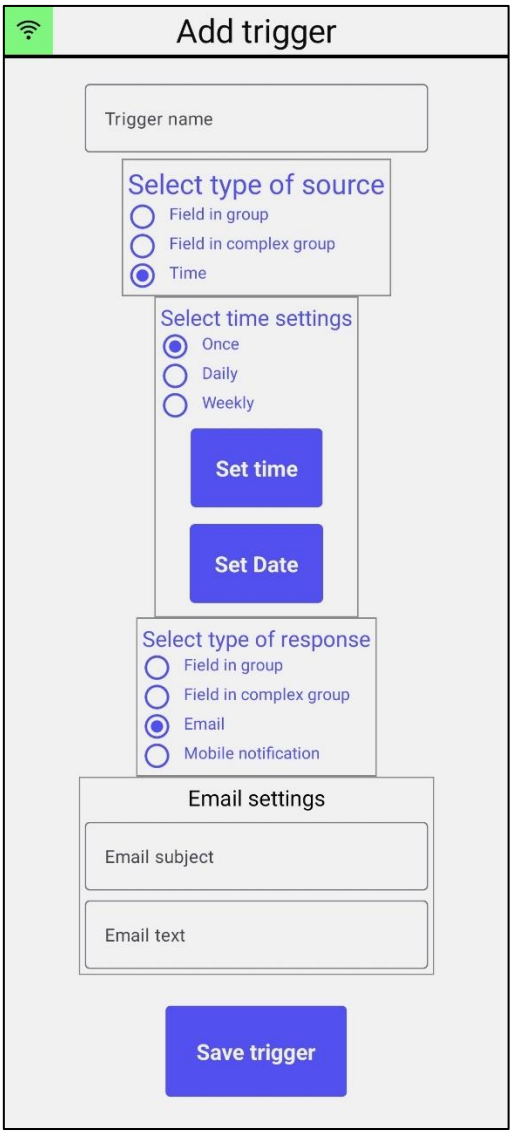
Pritiskom na tipku „Trigger settings“ na glavnom ekranu postavki dolazi se do postavki okidača. Ekran za upravljanje okidačima je na slici 3.71.



Slika 3.71. Izgled ekrana za postavke okidača

Sa tog ekrana moguće je navigirati se na ekran za dodavanje okidača te ekran za pregled postojećih okidača.

Ekran za dodavanje okidača može imati više izgleda ovisno o vrsti inicijacije okidača, vrsti reakcije okidača te stadiju do kojeg je korisnik došao s ispunom tih podataka. Primjeri izgleda ekrana su na slici 3.72. i prilogu 6.



Slika 3.72. Primjer izgleda ekrana za dodavanje okidača

Prvo korisnik mora unijeti naziv okidača. Potom korisnik mora unijeti vrstu inicijacije okidača koja može biti jedna od 3 vrste: okidač na vrijednost polja u grupi, okidač na vrijednost polja u kompleksnoj grupi te vremenski tip okidača. Ako je odabrana prva ili druga opcija potrebno je unijeti adresu polja na način da se prvo odabere željeni uređaj, nakon koje se može odabrati željena grupa ili kompleksna grupa u obliku padajućeg izbornika slično kao i pri dodavanju dopuštenja korisnicima. Nakon odabira adrese inicijacijskog polja potrebno je navesti uvjet okidanja.

Za brojčano polje se mora navesti vrijednost okidanja i tip okidanja, odnosno dali će se okidač okinuti ako je vrijednost jednaka, veća ili manja od neke vrijednosti, ili unutar ili izvan zadanog intervala vrijednosti.

Ako je polje tekstualno onda okidač također može imati 5 vrsta okidanja ovisno o vrijednosti – ako vrijednost polja počinje sa, završava sa, sadrži, jednako je ili nije jednako zadanom stringu. Potrebno je odabrati jednu od tih 5 tipova okidanja te tekstualnu vrijednost za okidač.

Ako je polje boolean tipa onda može okidati ako je vrijednost „true“ ili ako je „false“.

Ako se radi o polju s višestrukim izborom tada okidač može okidati ako je vrijednost polja jednako ili nije jednako zadanom stanju.

Ako se radi o RGB polju onda je postavljanje ovih parametara slično kao i kod brojčanog polja odnosno postoji 5 vrsta okidanja te jedna vrijednost granice ili dvije vrijednosti intervala ali postoji i kontekst za koji se mora odabrati jedna od 3 vrijednosti „Red“, „Green“ ili „Blue“ što označava za koju vrijednost se okidač veže.

Ako je odabran vremenski tip okidača onda je potrebno odabrati jedan od 3 tipa ponavljanja okidača – jednom, dnevno ili tjedno. Potrebno je izabrati i vrijeme te datum okidanja pomoću skočnih izbornika za datum i vrijeme.

S tim je pokriveno postavljanje inicijacije okidača te korisnik može prijeći na postavljanje reakcije okidača. Postoje 4 vrste reakcije okidača: postavljanje vrijednosti polja unutar grupe, postavljanje vrijednosti polja unutar kompleksne grupe, slanje elektroničke pošte te slanje mobilne notifikacije.

Ako se kao reakciju okidača odabere postavljanje vrijednosti polja potrebno je postaviti adresu reakcijskog polja i vrijednost na koju se želi postaviti.

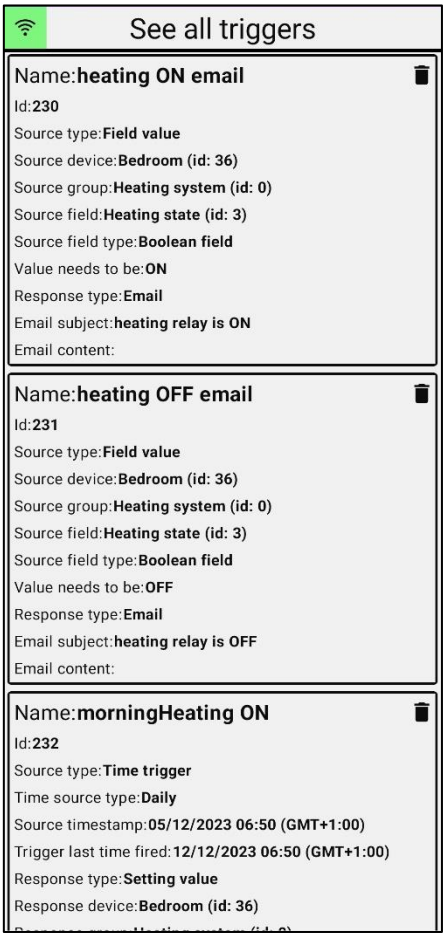
Kao polje inicijacije se može navesti bilo koje polje, a kao polje reakcije može navesti samo polja na koje ima pravo čitanja i pisanja i koja su ulazno izlazna polja..

Ako je odabrano slanje elektroničke pošte potrebno je unijeti naslov i sadržaj mail-a. Ovo je moguće izabrati samo ako korisnik ima potvrđenu email adresu. Ako je korisnik odabrao slanje mobilne notifikacije onda je potrebno unijeti naslov i sadržaj mobilne notifikacije.

Ukoliko je sve ispravno navedeno, nakon pritiska tipke za spremanje okidača prikazati će se toast poruka da je okidač uspješno pohranjen u bazu podataka.

Ako se sa ekrana za postavke okidača navigira na ekran za pregled postojećih okidača moguće je pregledati sve okidače te ih pojedinačno brisati.

Izgled ekrana za pregled okidača prikazan je na slici 3.73.



Slika 3.73. Izgled ekrana za pregled okidača

Svaki od navedenih ekrana je dostupan i u tamnom načinu rada, primjeri izgleda ekrana u tamnom načinu su prikazani u prilogu 7.

3.4. Mikro upravljačka komponenta

Cijela dosadašnja priča se svodi na to da nekakav fizički uređaj može od poslužitelja dobivati informacije koje mu omogućavaju da mijenja stanje na svojim I/O portovima.

Minimalni zahtjevi od mikro kontrolera su da se može spojiti na web, ostvarivati HTTP zahtjeve te stvoriti websocket veze. Zbog tih sitnih zahtjeva, puno mikro kontrolera i mini računala ispunjavaju zahtjeve, samostalno ili sa dodatnim komponentama. Zbog jednostavnosti bilo bi dobro iskoristiti uređaj koji ima ugrađen mrežni modul. Također je potrebno da uređaj ima dovoljno I/O pinova te mogućnosti povezivanja na razne module da bi se zadovoljile pojedinačne potrebe.

Prva zadaća uređaja nakon pokretanja je registracija na platformu. Primjer HTTP poziva za registraciju se nalazi u Prilogu 1. Iz registracije poslužitelj saznaje kakva je struktura uređaja. Uređaj se registrira na platformu koristeći svoj tajni ključ (device key) koji mora biti u potpunosti jedinstven na platformi. Da bi registracija bila uspješna, jedan od korisnika mora dodati uređaj sa tim tajnim ključem. Postupak je naveden u potpoglavlju 3.3.3. Tek nakon registracije administrator i/ili ostali korisnici mogu vidjeti koje I/O kontrole uređaj ima. Pri svakom idućem paljenju registracija podataka služi samo da uređaj potvrdi poslužitelju svoju strukturu, ostavljajući mogućnost da se uređaj reprogramira sa drugačijim I/O. U tom slučaju poslužitelj će se pobrinuti da se sva dopuštenja i okidači po potrebi obrišu (ako je polje sa identifikacijskim brojem #10 obrisano, onda se brišu sva dopuštenja i okidači na polje sa identifikacijskim brojem #10).

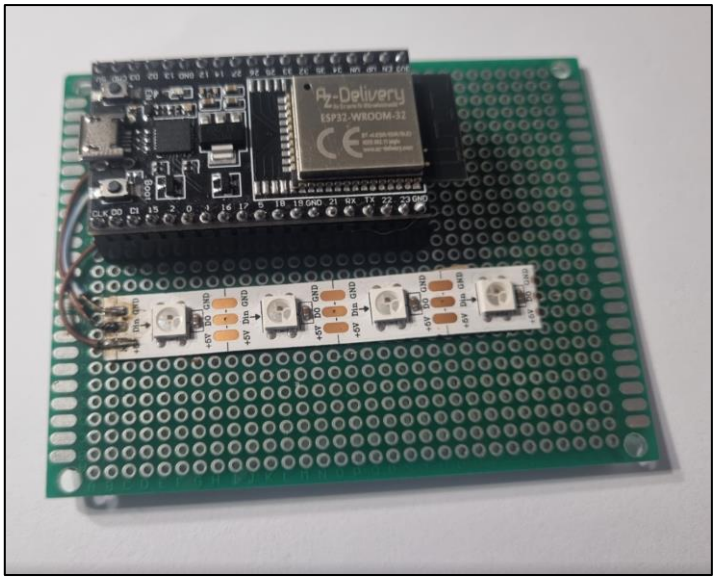
Nakon registracije uređaj se treba spojiti websocket vezom na poslužitelj kako bi primao poruke o promjeni stanja. Na uređaju mora biti implementirano čitanje poruke te refleksiju tih informacija na stanje I/O.

Uređaj također mora pratiti svoje ulaze kako bi na njihovu promjenu obavijestio poslužitelj HTTP pozivom prikazanim na slici 3.18.

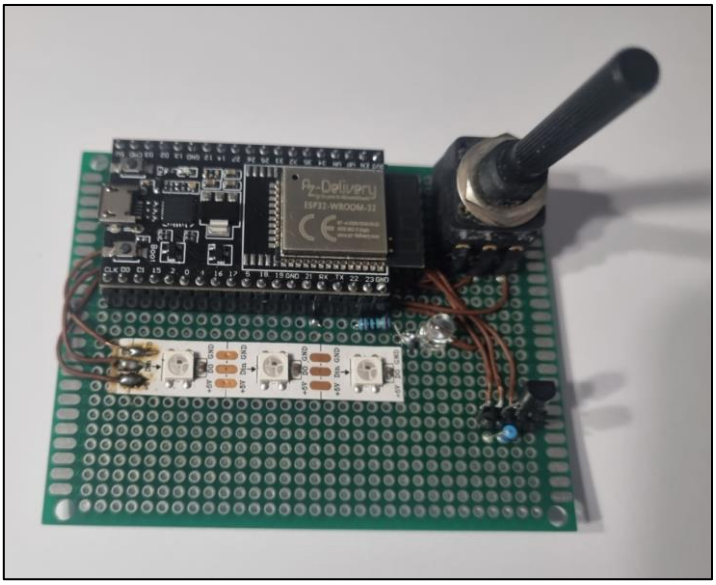
Sve navedeno je implementirano u C++ za mikrokontroler ESP32 koristeći materijale sa laboratorijskih vježbi kolegija Internet Objekata [8].

4. Testiranje predloženog rješenja

Za testiranje platforme izradio sam dva prototipa uređaja vidljiva na slikama 4.1. i 4.2.



Slika 4.1. Prvi prototip uređaja za testiranje sustava



Slika 4.2. Drugi prototip uređaja za testiranje uređaja

Prvi prototip se sastoji samo od 4 WS2812b LE diode. Dok se drugi sastoji od 3 takve te ima još i jednu običnu LE-diodu te potenciometar te temperaturni senzor. Takav I/O je dovoljan da se testiraju sve značajke platforme.

4.1. Testiranje postavki korisničkog računa

Prvi prvom otvaranju aplikacije korisniku se otvori forma za registraciju. Nakon što korisnik unese korisničko ime jedinstveno u sustavu i lozinku registracija će biti uspješna. Tada se korisnik

navigira na glavni ekran aplikacije koji je u tom trenutku prazan jer novi korisnik nema prava na nijedan uređaj.

Ako je korisnik pri registraciji naveo email adresu onda će za koju minutu primi email za potvrdu email adrese. Dogodi se da takav mail često odlazi u neželjenu poštu. Svaki idući put kada korisnik otvori aplikaciju bit će automatski ulogiran pomoću tokena koji je spremljen na memoriju telefona.

Korisnik se može odjaviti odlazeći u postavke korisničkog računa. Može odjaviti samo svoju instancu korisničkog računa ili sve instance svog korisničkog računa. Odjava svih instanci se izvrši unutar 2 sekunde. Nakon odjave se može ponovno prijaviti koristeći korisničko ime i lozinku.

Ako korisnik nije naveo email pri registraciji može ga kasnije dodati u korisničkim postavkama.

Korisnik može promijeniti lozinku svog korisničkog računa u korisničkim postavkama, ako zna trenutnu. Ako ne zna trenutnu može pri prijavi ispuniti formu za oporavak lozinke, potrebno je da ima potvrđenu email adresu s tim računom.

Naposljetku, korisnik može obrisati svoj korisnički račun – ako nije administrator nijednog uređaja, a ako jest administrator tada mora svoje ovlasti prebaciti na drugog korisnika ili obrisati te uređaje.

4.2. Testiranje postavljanja uređaja

Odlaskom na admin panel korisnik može dodati uređaj unoseći ime uređaja i novi tajni ključ za taj uređaj koji mora biti uprogramiran u samom uređaju.

Nakon što je uređaj dodan u sustav može upaliti isprogramiran uređaj sa nekakvim I/O na svojim pinovima. Uređaj tada ispravno registrira svoje podatke sa poslužiteljem i administrator tog uređaja „vidi“ sve kontrole koje uređaj nudi.

4.3. Testiranje upravljanja uređajem

Koristeći aplikaciju korisnik može upravljati svojim uređajima i ta promjena se ispod 2 sekunde reflektira na samom uređaju i na svim ostalim instancama aplikacije (ako korisnik ima pravo na taj uređaj).

4.4. Testiranje dodavanja dopuštenja

Administratori uređaja odlaskom na postavke dopuštenja mogu dodati dopuštenja na svoj uređaj ili dijelove uređaja. U trenutku kada dodaju dopuštenje nekom korisniku, taj korisnik u roku 2 sekunde može vidjeti taj uređaj i početi s njim upravljati ovisno o tipu dopuštenja.

4.5. Testiranja okidača

Okidači su testirani u svim kombinacijama, postoje 3 načina inicijacije i 4 načina reakcije s tim da su 2 od 3 načina inicijacije i 2 od 4 načina reakcija vezani za polja unutar uređaja i stoga postoji 5 načina njihovih postavljanja (5 vrsta polja). Sve je istestirano i pronađena je samo jedna mana, a ta je da mobilne notifikacije ne dolaze ukoliko korisnik ne upali aplikaciju barem jedanput dnevno, razlog je vjerojatno ažuriranje tokena.

4.6. Cijena rada sustava

Cijena rada samog poslužitelja je ovisna o kojem hostingu se radi, ako se koristi Render hosting kako je to navedeno u potpoglavlju 3.2.7. onda je besplatno, a ako se koristi Heroku hosting onda je 7€ mjesečno. Jedino što se osim toga naplaćuje je Firestore baza podataka.

Firestore baza podataka ima besplatni plan, no u nekim slučajevima on može biti nedovoljan.

Firestore se naplaćuje u 4 glavna segmenta: po količini čitanja iz baze, količini pisanja u bazu, količini brisanja iz baze i po količini spremljenih podataka. Za svaku od ovih parametara postoji besplatna kvota, a iznad nje se naplaćuje po informacijama sa Firestore stranicama [8]. Cijene također ovise o mjestu na kojima je ta baza hostana. Podaci navedeni u tablici 4.1. su za Frankfurt jer je to najbliže mjestu na kojoj je hostan poslužitelj (također Frankfurt).

Tablica 4.1. Tablica sa izračunima prosječne mjesečne cijene Firestore baze podataka

	Besplatna kvota	Obračunska jedinica	Cijena po obračunskoj jedinici
Čitanje iz baze	50.000	100.000	$0.039
Pisanje u bazu	20.000	100.000	$0.117
Brisanje iz baze	20.000	100.000	$0.013
Spremljeni podaci	1 GiB	GiB mjesečno	$0.117

Važno je identificirati da je ponašanje uređaja i korisnika ono koje određuje cijenu hostanja baze podataka. Ako će korisnik svakih par sekundi mijenjati stanje uređaja, to će povećati broj čitanja i pisanja u bazi stoga će mjesečna cijena biti znatno veća. Osim takvih neobičnih korištenja platforme od strane korisnika, glavni izvor opterećenja na bazu su senzori spojeni na uređaje koji kontinuirane motre nekakve analogne veličine poput temperature prostorije, vlažnosti zraka, koncentracija ugljičnog dioksida u prostoriji i slično.

Iz načina na koji je poslužitelj programiran imamo oko 10 čitanja iz baze za svaku promjenu stanja uređaja. Razlog za tako velik broj je autentikacija uređaja, provjera postoji li polje koje uređaj želi promijeniti te razne provjere mora li se ta nova informacija proslijediti do nekih trenutno aktivnih korisnika ili mora li se okinuti nekakav okidač. Poslužitelj u toku zahtjeva za promjenom stanja uređaja samo jedanput zapisuje podatke u bazu (promijeni vrijednost).

Možemo na primjeru izračunati mjesečnu cijenu Firestore-a. Uzmimo da platforma ima 30 uređaja koji su cijelo vrijeme spojeni na poslužitelj i da svaki motri ukupno 10 analognih senzora koji za prihvatljiv rad šalju informacije na poslužitelj svakih 5 minuta. Dobijemo da je to ukupno 86.400 pojedinačnih zahtijeva u bazu odnosno 864.000 čitanja u bazu podataka i 86.400 pisanja u bazu podataka. Koristeći informacije iz tablice 4.1. možemo izračunati mjesečnu potrošnju. Rezultati su prikazani u tablici 4.2, vidimo da je konačna mjesečna cijena za ovaj primjer iznosi $14.04.

Tablica 4.2. Izračun mjesečnog troška Firestore baze podataka

	Broj potrebnih jedinica	Potrebne obračunske jedinice	Dnevna cijena	Mjesečna cijena
Čitanje iz baze	864.000	9	$0.351	$10.53
Pisanje u bazu	86.400	1	$0.117	$3.51
Brisanje iz baze	<20.000	0	$0	$0
Spremljeni podaci	<1GiB	0	$0	$0

4.7. Utvrđene mane i potencijalna poboljšanja

Najveća utvrđena mana platforme je nestabilnost mobilnih notifikacija i odlazak email-a u neželjenu poštu.

Potencijalna poboljšanja platforme su:

1) Razvijanje estetike korisničkog sučelja
2) Praćenje prošlih vrijednosti stanja uređaja
3) Korisnički definirani pogledi, korisnici bi mogli iz aplikacije pregledavati različite dijelove uređaja odjednom, slično kao u MyDevices platformi.
4) Proširenje aplikacije na wearOS i web sučelje

5. Zaključak

Zadatak diplomskog rada bio je proučiti komercijalno dostupne platforme za nadzor i upravljanje uređajima te izraditi vlastitu. U Node.js okruženju sam razvio poslužitelj koji obavlja sve potrebne funkcionalnosti koje su potrebne kako bi se razni IoT uređaji mogli na njega spojiti te slati podatke o svojoj konfiguraciji i trenutnom stanju. Korisnik upravlja sa podacima na poslužitelju odnosno stanjem uređaja preko android aplikacije razvijene u android studiju koristeći Kotlin i Jetpack compose alat. Korisnik također može preko aplikacije dodavati i brisati uređaje i dodavati dopuštenja drugim korisnicima na uređaje koje je on dodao u sustav. Omogućeno je upravljanje korisničkim računom u vidu oporavka računa pomoću email-a i prisilnog odjavljivanja sa svih instanci korisničkog računa. Platforma ima omogućene okidače koji omogućavaju korisnicima automatsko upravljanje uređajem i dojavu mobilnom notifikacijom ili emailom ako dođe do pred definirane promjene stanja uređaja.

LITERATURA

[1] MyDevices Cayenne dokumentacija https://community.mydevices.com/t/cayenne-documentation/150 [10.9.2023.]

[2] Blynk dokumentacija https://docs.blynk.io/en/ [10.9.2023.]

[3] NodeJs dokumentacija https://nodejs.org/en/docs [10.9.2023.]

[4] Firebase dokumentacija https://firebase.google.com/docs/firestore [10.9.2023.]

[5] Heroku hosting dokumentacija https://devcenter.heroku.com/categories/reference [10.9.2023.]

[6] Render hosting dokumentacija https://render.com/docs [10.9.2023.]

[7] Android dokumentacija https://developer.android.com/docs [10.9.2023.]

[8] Laboratorijska vježba 7, Internet objekata, FERIT

[9] Firestore cjenik https://cloud.google.com/firestore/pricing [10.9.2023.]

SAŽETAK

U diplomskom radu su opisani neki od komercijalno dostupnih platformi za upravljanje i nadzor uređaja te je opisana izrada vlastite platforme. Objašnjena je arhitektura sustava, dostupni tipovi podataka za upravljanje uređajima i osnovna konfiguracija koji uređaji moraju slijediti kako bi bili kompatibilni sa sustavom. Poslužitelj je opisan kroz HTTP zahtjeve koje zaprima i websocket poruke koje odašilje korisnicima i uređajima. Android aplikacija je prikazana sa objašnjenjima i slikama svakog ekrana, objašnjeno je kako korisnici mogu upravljati uređajima, dopuštenjima uređaja, svojim korisničkim računom te okidačima.

Ključne riječi: Android, Internet stvari, Firestore, Node.js, Websocket.

ABSTRACT

Platform for remote monitoring and control of devices

This master's thesis describes some commercially available platforms for device monitoring and control and describes the development of my own platform. The system architecture is explained, along with the available types of data for device management and the basic configuration that devices must comply with to in order to be compatible with the system. The server is described through HTTP requests it receives and WebSocket messages it sends to users and devices. The Android application is presented with explanations and images of each screen, and it is explained how users can manage devices, device permissions, their user account, and triggers.

Key words: Android, Internet of things, Firestore, Node.js, Websocket.

ŽIVOTOPIS

Kristian Klišković rođen je 15. siječnja 2000. godine u Slavonskom Brodu. Pohađao je osnovnu školu Vladimir Nazor u Slavonskom Brodu. 2015. godine upisuje Tehničku školu Slavonski Brod. U srednjoj školi otkriva interes u poljima elektrotehnike i računarstva, ponajviše za elektroniku i programiranje. U Tehničkoj školi sudjeluje na deset županijskih i državnih natjecanja iz Matematike, Fizike i Elektrotehnike, u drugom razredu osvaja peto mjesto na državnom natjecanju iz Osnova elektrotehnike i Mjerenja u elektrotehnici. Sudjeluje i na 5 smotri radova i izložba inovacija. Osvaja brončanu plaketu na Izložbi inovacija u Ivanić Gradu i dvije zlatne medalje na izložbi inovacija mladi@inovacije za radove „Arduino Križić-kružić“ i „Digitalni sat“. 2018. godine upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer Elektrotehnika i informacijska tehnologija gdje 2020. godine dobiva priznanje za uspješnost u studiranju radi prosjeka 5.0 u prva tri semestra. 2021. godine diplomira na temu „Sustav za praćenje automobila“ i upisuje diplomski studij Elektrotehnika, smjer komunikacije i informatika na FERIT-u.

PRILOZI

Prilog 1: Primjer tijela HTTP poziva za konfiguraciju strukture uređaja

```
{
  "deviceKey": "secretKey",
  "deviceFieldGroups": [
    {
      "id": 0,
      "groupName": "group1_rename",
      "fields": [
        {
          "id": 1,
          "fieldName": "field1",
          "fieldType": "numeric",
          "fieldValue": {
            "fieldValue": -1,
            "minValue": -1,
            "maxValue": 25,
            "valueStep": 1,
            "fieldDirection": "input"
          }
        },
        {
          "id": 2,
          "fieldName": "field2",
          "fieldType": "text",
          "fieldValue": {
            "fieldValue": "23 deg C",
            "fieldDirection": "input"
          }
        }
      ]
    }
  ],
  "deviceFieldComplexGroups": [
    {
      "id": 0,
      "groupName": "complexGroup1",
      "currentState": 0,
      "fieldGroupStates": [

        {
          "id": 1,
          "stateName": "rgb",
          "fields": [
            {
              "id": 0,
              "fieldName": "RGB",
              "fieldType": "RGB",
              "fieldValue": {
                "R": 0,
                "G": 0,
                "B": 0,
                "fieldDirection": "input"
              }
            }
          ]
        },
        {
          "id": 2,
```

```
          "stateName": "animations",
          "fields": [
            {
              "id": 0,
              "fieldName": "animations",
              "fieldType": "multipleChoice",
              "fieldValue": {
                "values": [
                  "OFF",
                  "A1",
                  "A2",
                  "A3",
                  "A4",
                  "A5"
                ],
                "fieldValue": 0,
                "fieldDirection": "input"
              }
            }
          ]
        }
      ]
    },
    {
      "id": 1,
      "groupName": "complexGroup2",
      "currentState": 0,
      "fieldGroupStates": [
        {
          "id": 0,
          "stateName": "individual",
          "fields": [
            {
              "id": 0,
              "fieldName": "RField",
              "fieldType": "numeric",
              "fieldValue": {
                "fieldValue": 0,
                "minValue": 0,
                "maxValue": 25,
                "valueStep": 1,
                "fieldDirection": "input"
              }
            }
          ]
        }
      ]
    }
  ]
}
```

Prilog 2: Primjer poruke koja se šalje uređaju nakon promjene njegovog stanja


```
{
  "messageType": "deviceData",
  "data": {
    "id": 3,
    "deviceKey": "key22",
    "deviceName": "Device1",
    "userAdminId": 2,
    "deviceFieldGroups": [
          {
            "id": 50,
            "fieldName": "field0",
            "fieldType": "numeric",
            "fieldValue": {
              "minValue": -1,
              "maxValue": 25,
              "valueStep": 1,
              "prefix": "T=",
              "sufix": "°C",
              "fieldDirection": "input",
              "fieldValue": 21
            },
            "readOnly": true
          },
          {
            "id": 99,
            "fieldName": "field9",
            "fieldType": "button",
            "fieldValue": {
              "fieldDirection": "input",
              "fieldValue": true
            },
            "readOnly": true
          }
        ]
      },
    ],
    "deviceFieldComplexGroups": [
      {
        "id": 0,
        "groupName": "complexGroup1",
        "currentState": 2,
        "fieldGroupStates": [
          {
            "id": 1,
            "stateName": "rgb",
            "fields": [
              {
                "fieldName": "RGB",
                "id": 0,
                "fieldType": "RGB",
                "fieldValue": {
                  "fieldDirection": "input",
                  "R": 246,
                  "G": 11,
                  "B": 151
                 }
              }
            ]
          },
```

```
          {
            "id": 2,
            "stateName": "animations",
            "fields": [
              {
                "fieldName": "animations",
                "id": 0,
                "fieldType": "multipleChoice",
                "fieldValue": {
                  "values": [
                    "OFF",
                    "A1",
                    "A2",
                    "A3",
                    "A4",
                    "A5"
                  ],
                  "fieldDirection": "input",
                  "fieldValue": 2
                }
              }
            ]
          }
        ],
        "readOnly": true
      },
    ],
    "updateTimeStamp": 1694280372370,
  }
}
```

Prilog 3: Primjer poruke koja se šalje korisniku nakon promjene stanja jednog od uređaja na koje ima pravo.


```
{
  "messageType": "deviceData",
  "data": [
    {
      "id": 3,
      "deviceKey": "key22",
      "deviceName": "Device1",
      "userAdminId": 2,
      "deviceFieldGroups": [
            {
              "id": 50,
              "fieldName": "field0",
              "fieldType": "numeric",
              "fieldValue": {
                "minValue": -1,
                "maxValue": 25,
                "valueStep": 1,
                "prefix": "T=",
                "sufix": "°C",
                "fieldDirection": "input",
                "fieldValue": 21
              },
              "readOnly": true
            },
            {
              "id": 99,
              "fieldName": "field9",
              "fieldType": "button",
              "fieldValue": {
                "fieldDirection": "input",
                "fieldValue": true
              },
              "readOnly": true
            }
          ]
        },
      ],
      "deviceFieldComplexGroups": [
        {
          "id": 0,
          "groupName": "complexGroup1",
          "currentState": 2,
          "fieldGroupStates": [
            {
              "id": 1,
              "stateName": "rgb",
              "fields": [
                {
                  "fieldName": "RGB",
                  "id": 0,
                  "fieldType": "RGB",
                  "fieldValue": {
                    "fieldDirection": "input",
                    "R": 246,
                    "G": 11,
                    "B": 151
                  }
                }
```

```
            ]
          },
          {
            "id": 2,
            "stateName": "animations",
            "fields": [
              {
                "fieldName": "animations",
                "id": 0,
                "fieldType": "multipleChoice",
                "fieldValue": {
                  "values": [
                    "OFF",
                    "A1",
                    "A2",
                    "A3",
                    "A4",
                    "A5"
                  ],
                  "fieldDirection": "input",
                  "fieldValue": 2
                }
              }
            ]
          }
        ],
        "readOnly": true
      },
    ],
    "updateTimeStamp": 1694280372370,
    "isActive": true
  }
 ]
}
```

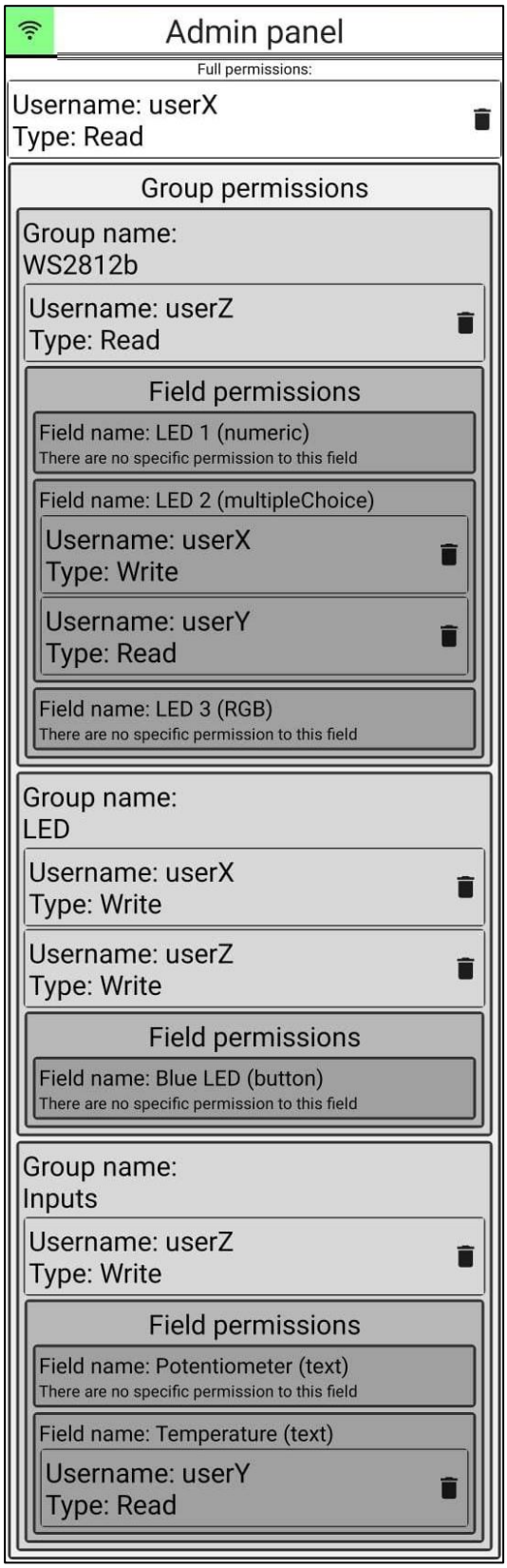
Prilog 4: Rezultat HTTP zahtjeva za dobivanje svih prava na uređaj

HTTP status kod: 200 OK

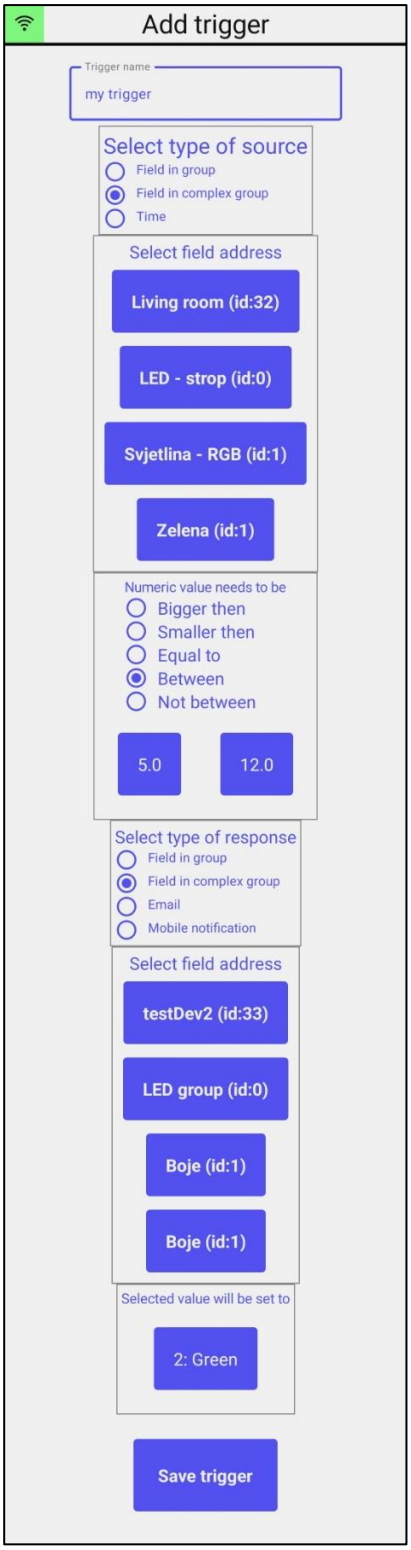
Tijelo odgovora:

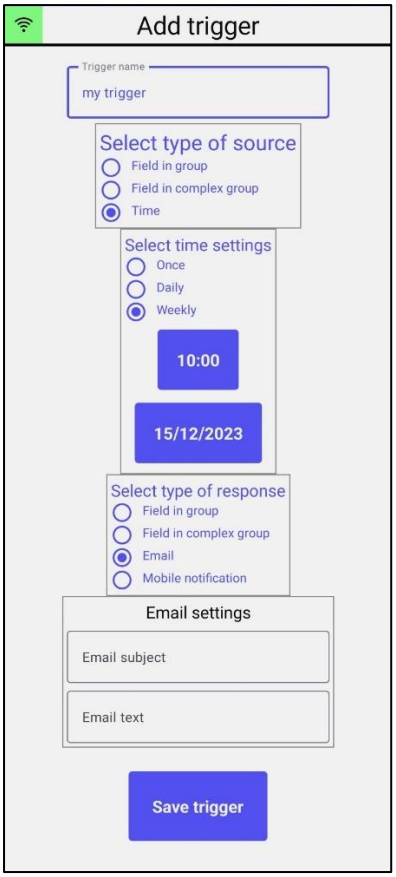

```
{
    "userPermissions": [
        {
            "userId": 1,
            "username": "Damir",
            "readOnly": true
        },
        {
            "userId": 23,
            "username": "Maris",
            "readOnly": true
        }
    ],
    "groups": [

        {
            "groupId": 1,
            "groupName": "group2",
            "fields": [
                {
                    "fieldId": 1,
                    "fieldType": "numeric",
                    "fieldName": "field1",
                    "userPermissions": [
                        {
                            "userId": 23,
                            "username": "Maris",
                            "readOnly": false
                        }
                    ]
                },
                {
                    "fieldId": 2,
                    "fieldType": "text",
                    "fieldName": "field2",
                    "userPermissions": [
                        {
                            "userId": 23,
                            "username": "Maris",
                            "readOnly": false
                        }
                    ]
                }
            ],
            "userPermissions": []
        }
    ],
    "complexGroups": [

        {
            "complexGroupId": 1,
            "complexGroupName": "complexGroup1",
            "userPermissions": [
                {
                    "userId": 23,
                    "username": "Maris",
                    "readOnly": false
                }
            ]
        }
    ]
}
```

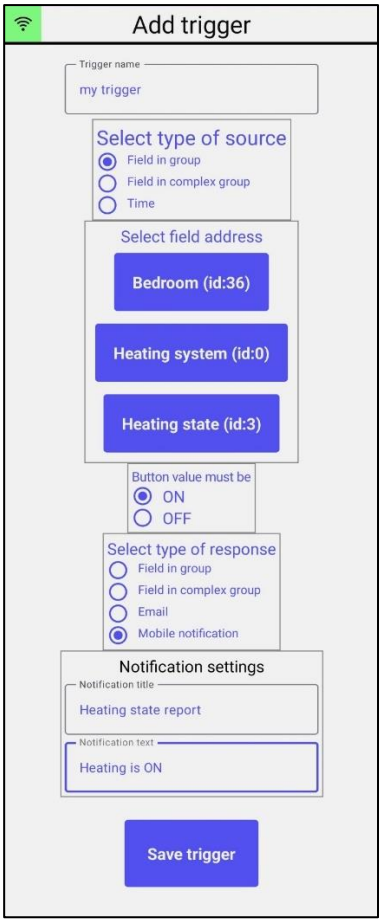
Prilog 5: Prikaz ekrana za prikaz svih dopuštenja na uređaj



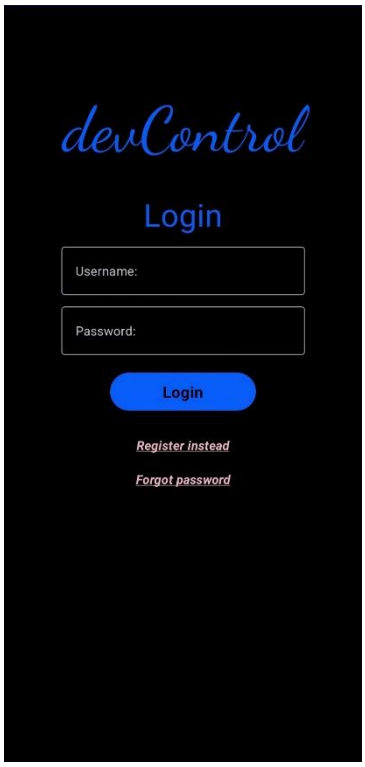
Prilog 6: Primjeri izgleda ekrana za dodavanje okidača

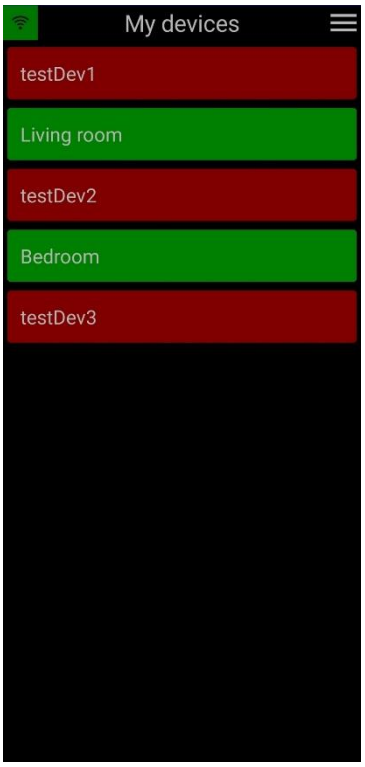






Prilog 7: Primjeri izgleda ekrana u tamnom načinu





Na CD-u:

1. Node.js projekt za poslužitelj
2. Android aplikacija
3. Android aplikacija (.apk)
4. Primjer firmwear-a uređaja