Optimizacija rada kućanskih uređaja s obzirom na fluktuaciju cijena električne energije

Cvijetić, Ivan

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1624

OPTIMIZACIJA RADA KUĆANSKIH UREĐAJA S OBZIROM NA FLUKTUACIJU CIJENA ELEKTRIČNE ENERGIJE

Ivan Cvijetić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1624

OPTIMIZACIJA RADA KUĆANSKIH UREĐAJA S OBZIROM NA FLUKTUACIJU CIJENA ELEKTRIČNE ENERGIJE

Ivan Cvijetić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1624

Pristupnik:	**Ivan Cvijetić (0036540320)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Hrvoje Pandžić

Zadatak: **Optimizacija rada kućanskih uređaja s obzirom na fluktuaciju cijena električne energije**

Opis zadatka:

Kućanstva predstavljaju izuzetan potencijal za pružanje odziva potrošnje električne energije, čime se otvara prostor i za efikasnije upravljanje energetskim resursima i smanjenje troškova električne energije. U skladu s tim, razvija se matematički model koji ima za cilj optimizaciju rada uređaja unutar kućanstava uzimajući u obzir fluktuacije tržišnih cijena električne energije. Korištenjem alata poput HomeAssistant, signali generirani ovim modelom bit će usmjereni prema uređajima radi prilagodbe njihovog rada trenutnim uvjetima na tržištu i potrebama elektroenergetskog sustava.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem mentoru prof. dr. sc. Hrvoju Pandžiću, univ. mag. ing. el. techn. inf. Nikolini Čović i mag. ing. Matku Mesaru na korisnim savjetima i smjernicama prilikom izrade rada, kao i obitelji na pruženoj podršci.

SADRŽAJ

1. Uvod

S obzirom na to da se neupravljivi obnovljivi izvori energije sve više integriraju u elektroenergetski sustav, aktivni potrošači imaju sve izraženiju ulogu u održavanju ravnoteže sustava. Iako je odziv potrošnje u kućanstvima u razvoju, još uvijek nije prisutan u mnogim zemljama. Glavni razlog tomu je kompleksnost pružanja odziva potrošnje u kućanstvima. Ono zahtijeva postavljanje uređaja koji podržavaju odziv potrošnje, koje većina kućanstava i dalje ne posjeduje. Također, potrebno je istodobno upravljati uređajima velikog broja kućanstava kako bi se postigao jednak utjecaj na elektroenergetsku mrežu kao što ima uvođenje odziva potrošnje u neko industrijsko postrojenje. Međutim, nedavne tehnološke inovacije i brzo usvajanje programa za upravljanje energijom te pametnih mreža otvaraju mogućnosti za uključivanje odziva potrošnje i u kućanstva.

U sklopu istraživanja mogućnosti odziva potrošnje, odnosno sudjelovanja pametnih kućanskih uređaja u uravnoteženju elektroenergetskog sustava, u laboratoriju na Zavodu za visoki napon i energetiku instaliran je pozamašan broj pametnih kućanskih uređaja, čija se potrošnja električne energije kontinuirano prati. Glavni zadatak je povezati sve uređaje na Home Assistant platformu, omogućiti njihovo daljinsko upravljanje te izraditi web aplikaciju kojom će se omogućiti rad uređaja po načelima odziva potrošnje uporabom optimizacijskog modela i Home Assistanta.

2. Odziv potrošnje

Odziv potrošnje (eng. Demand Response) [1] odnosi se na uravnoteženje proizvodnje i potrošnje u električnim mrežama poticanjem kupaca da prilagode svoju potrošnju električne energije na vrijeme kada je opterećenje mreže nisko, obično putem cijena ili novčanih poticaja. Uz pametne mreže i pohranu energije, odziv potrošnje važan je izvor fleksibilnosti kojim se osigurava stabilnost i pouzdanost elektroenergetskih mreža u vremenu rastućeg udjela varijabilnih obnovljivih izvora energije i povećanog opterećenja sustava.

U borbi protiv globalnog zatopljenja IEA (International Energy Agency) osmislila je Net Zero scenarij kojim se želi postići da se do 2050. godine smanje emisije stakleničkih plinova u atmosferu i poveća uklanjanje istih. Upravo zato doći će do elektrifikacije sektora kao što su prijevoz i grijanje kućanstava kao i opsežne primjene solarnih fotonaponskih sustava i energije vjetra što će uzrokovati značajni rast potrošnje električne energije. Međutim, budući da količina dobivene energije od Sunca i vjetra ovisi o vremenskim uvjetima i dobu dana, stvaraju se dodatni izazovi za elektroenergetsku mrežu. Za rješavanje ovih izazova i minimiziranje zahtjeva za novom, skupom prijenosnom i distribucijskom infrastrukturom, tehnologije poput odziva potrošnje igraju ključnu ulogu. Odziv potrošnje omogućuje mreži učinkovito upravljanje varijabilnošću navedenih izvora energije prilagođavanjem obrazaca potrošnje električne energije. Korištenjem odziva potrošnje, električna mreža može se bolje prilagoditi proizvodnji obnovljivih izvora energije i optimizirati ukupnu učinkovitost sustava.
Nove tehnologije mogu pomoći u automatiziranju odziva potrošnje putem pametnih međusobno povezanih uređaja. Na primjer, pametni termostati mogu automatski podešavati temperaturu u kućama na temelju trenutne vanjske temperature, dok pametni punjači za električna vozila mogu odgoditi punjenje do trenutka kada je dostupan višak energije iz obnovljivih izvora. Slično tome, kućanski aparati poput perilica rublja ili suđa mogu biti programirani da se uključuju tijekom perioda niskog opterećenja sustava ili kada su cijene električne energije niže.

3. Opis matematičkog modela

Kreiran je matematički model kućanstva koji sudjeluje u programu odziva potrošnje paljenjem i gašenjem svojih kućanskih uređaja. Optimizacijskim postupkom nad matematičkim modelom danom u (3.1)-(3.2) određuju se periodi u kojima će kućanski uređaji raditi, tako da se smanje troškovi električne energije, ali uz poštivanje tehničkih ograničenja. Funkcija cilja koristi podatke o cijenama električne energije i podatak o vremenu rada svakog uređaja u kućanstvu koji je korisnik dužan unijeti. Model zatim određuje optimalne termine rada uređaja kako bi se troškovi minimizirali, uzimajući u obzir varijabilnost cijena tijekom dana. Tako će uređaji raditi u satima kada su cijene energije niže, odnosno kad je elektroenergetska mreža manje opterećena, što rezultira manjim ukupnim troškovima za kućanstvo i u isto vrijeme rasterećenju elektroenergetske mreže.

Funkcija cilja:

$$\min_{x[d][i]} \quad \sum_{d \in \text{Devices}} \sum_{i=0}^{23} c[i] \cdot x[d][i] \cdot p \tag{3.1}$$

Ograničenja:

$$\forall d \in \text{Devices} : \sum_{d \in \text{Devices}} x[d] \geq s \tag{3.2}$$

gdje je:
$s \in \{0, 1, 2, ..., 23\}$ - broj sati koji želimo da uređaj je uređaj uključen,
$d \in$ Devices - skup svih uređaja,
$i \in \{0, 1, 2, ..., 23\}$ - sati u danu,
$c[i]$ - cijena električne energije u i-tom satu,
$x[d][i]$ - binarna varijabla koja predstavlja je li uređaj d uključen u i-tom satu.
p - snaga koju uređaj koristi za rad; za potrebe demonstracije i zbog jednostavnosti iznosit će 1kW za svaki uređaj (izlaz modela bit će analogan kao i kad bi se primijenile stvarne vrijednosti snaga uređaja)

Ovaj model rezultira rasporedom rada uređaja po satima.

3.1. Implementacija

Podatke o cijenama električne energije preuzeo sam sa stranice HEP-a [2]. Koristio sam podatke crvenog tarifnog modela koji uključuju troškove distribucijske i prenose mreže te posebnu naknadu za poticanje proizvodnje električne energije iz obnovljivih izvora.
Matematički model implementirao sam u Pythonu koristeći Gurobi [3] i Pandas [4] biblioteke.

3.1.1. Opis koda

Implementacija svih potrebnih funkcionalnosti zahtijevala je izradu funkcije **optimizeElectricityPrices**. Ona prima dva parametra, od koji je jedan putanja do Excel datoteke s podacima o cijenama električne energije, a drugi rječnik u kojem su imena pametnih uređaja ključevi,a vrijednosti su broj sati rada uređaja. Za učitavanje podataka iz Excel datoteke potrebno je koristiti funkciju **read_excel** iz Padnas biblioteke koja kao argumente prima putanju do datoteke i ime lista(engl. sheet) s kojeg je potrebno učitati podatke. Zatim dobiven okvir pretvoren je u NumPy polje, što u Pythonu označava matrični zapis (Slika 3.1).

```
import pandas as pd
from gurobipy import *

def optimizeElectricityPrices(filePath,optimizationDevices:dict):
    data = pd.read_excel(filePath, sheet_name='Sheet1')
    pricesOfElectricityByHour = data.to_numpy()[:,1]
```

Slika 3.1: Čitanje podataka o cijenama električne energije

Nakon toga potrebno je u nastavku **optimizeElectricityPrices** izraditi varijable koje će koristiti funkcija cilja našeg modela i samu funkciju cilja.
Varijable su spremljene u rječnik **device_vars**, koji će za svaki uređaj iz rječnika **pricesOfElectricityByHour**, u sebi sadržavati ime uređaja kao ključ, a kao vrijednost će imati specijalan rječnik koji se kreira kad se nad modelom pozove funkcija **addVars**. Specijalni će rječnik kao ključeve imati poziciju elementa u rječniku, a vrijednost će

mu biti gurobi binarna varijabla koja će označavati je li uređaj upaljen(1) ili ugašen(0) (Slika 3.2). Specijalni rječnik će imati 24 zapisa gdje svaki zapis označava jedan sat u danu.

Struktura specijalnog rječnika:

```
{
    0: <gurobi.Var device1[0]>,
    1: <gurobi.Var device1[1]>,
    2: <gurobi.Var device1[2]>,
    ...
    23: <gurobi.Var device1[23]>
}
```

```
time = 24

m = Model('Electricity prices')

device_vars = {device: m.addVars(24,vtype=GRB.BINARY, name=device) for device in optimizationDevices}
```

Slika 3.2: Kreiranje matematičkog modela

Za implementaciju funkcije cilja koristio sam python funkciju **quicksum**. Zbrajaju se umnošci cijena električne energije u određenom satu, vrijednosti gurobi binarnih varijabli za taj sat i snage uređaja te se to provodi za svaki uređaj. Kreiranu funkciju cilja predajemo modelu pozivom funkcije **setObjective** nad modelom (Slika 3.3).

```
total_cost=quicksum(
pricesOfElectricityByHour[i]*device_vars[device][i] for device in optimizationDevices for i in range(time)
)
m.setObjective(total_cost)
```

Slika 3.3: Kreiranje funkcije cilja

Ograničenja na varijable zadaju se pozivom funkcije **addConstr** nad modelom. Kao ograničenje zadano je da zbroj vrijednosti svih gurobi binarnih varijabli za uređaj treba biti veći ili jednak broju sati rada za taj uređaj. Nakon toga se poziva funkcija **optimize** nad modelom čime započinje postupak optimizacije (Slika 3.4).

```
for device in optimizationDevices:
    m.addConstr(quicksum(device_vars[device][i] for i in range(time)) >= optimizationDevices[device])

m.optimize()
```

Slika 3.4: Dodavanje ograničenja varijablama modela

Model će izračunati minimum od funkcije cilja tako što će u gurobi varijable za svaki uređaj upisati jedan ili nula ovisno o tome koje će kombinacije vrijednosti gurobi varijabli za svaki uređaj dati najmanju vrijednost funkcije cilja. Rezultati optimizacije, odnosno podaci kad koji uređaj treba raditi se upisuju u rječnik **result** koji kao vrijednost ključa prima naziv uređaja, a kao vrijednost raspored rada uređaja sljedećeg dana po satima (Slika 3.5). Naveden rezultantni rječnik će u kombinaciji s Home Assistant API pozivima omogućiti upravljanje uređajima.

```
result={device: [int(device_vars[device][i].x) for i in range(time)] for device in optimizationDevices}
return(result)
```

Slika 3.5: Spremanje rezultata modela

4. Home Assistant

4.1. Uvod

Home Assistant [5] je besplatan program otvorenog koda za kućnu automatizaciju dizajniran kao integracijska platforma neovisna o ekosustavu interneta stvari (engl. Internet Of Things) [6] i pametno kućno središte za kontrolu pametnih kućnih uređaja, s fokusom na lokalnu kontrolu i privatnost. Njegovom se sučelju može pristupiti putem web korisničkog sučelja ili putem njihove mobilne aplikacije.

4.2. Instalacija

Sustav kućne automatizacije trebao bi postati dio svakog kućanstva što Home Assistant omogućuje svojim različitim načinima instalacije. Home Assistant Green, koji je moguće kupiti na Home Assistant web stranici, je malo računalo opće namjene na koji je već instaliran Home Assistant operacijski sustav te je spreman za korištenje putem njegovog web sučelja. Ova opcija instalacije povećala je popularnost Home Assistanta jer i osobama koji se ne žele baviti konfiguriranjem sustava omogućuje korištenje po relativno niskoj cijeni. Također, moguće je instalirati Home Assistant na bilo kojem računalu opće namjene npr. Raspberry Pi, kao i na virtualnoj mašini, te je čak dostupan kao rješenje na spremniku (engl. container), koje bi se moglo izvoditi na Dockeru. U laboratoriju za odziv potrošnje na ZVNE-u, Home Assistant je instaliran na uređaju Raspberry Pi.

4.3. Povezivanje uređaja

Nakon kreiranja vlastitog korisničkog računa i prijave u sustav preko web sučelja moguće je početi s povezivanjem uređaja. Home Assistant dokumentacija navodi da je svaki uređaj spojen na Home Assistant različit i potrebne su osnovne komponente za

njihovu prezentaciju. Osnovne komponente za dodavanje i upravljanje uređajima i uslugama su integracije i stanja. Svaka od ovih osnovnih komponenti ima svoju specifičnu funkciju.
Integracije su osnovne komponente koje predstavljaju sve entitete kao što su uređaji, usluge i događaji te njima upravlja sustav. Sustav može pristupiti funkcionalnosti uređaja ili usluga putem kanala koji svaka odgovarajuća integracija sadrži.
Stanje se odnosi na trenutačno stanje nekog entiteta. Svaki entitet ima jedno glavno stanje koje može biti različitog tipa, ovisno o vrsti entiteta. Primjeri stanja uključuju "on" ili "off" za prekidače, "home" ili "away" za praćenje prisutnosti, te specifične vrijednosti za senzore (npr. temperatura, vlažnost). Svaki entitet ima stanje koje se sastoji od glavne vrijednosti i dodatnih atributa. Glavna vrijednost stanja može biti string, broj ili neki drugi tip podatka, dok atributi pružaju dodatne informacije o entitetu.

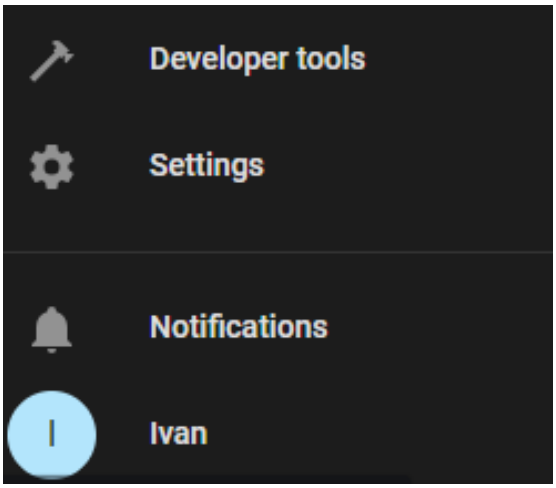
Svaki uređaj zahtijeva svoj način integracije u Home Assistant [7]. Određeni uređaji mogli su se izravno povezati s Home Assistantom, neki su se integrirali s pomoću njihovih službenih aplikacija, dok ostali nisu bili podržani od strane Home Assistanta, pa su korisnici pronašli način putem prilagođenog programiranja. Prilagođena integracija je moguća ako je proizvođač razvio otvoreno sučelje za programiranje aplikacija, poznato kao Application Programming Interface (API).

Kao primjer spajanja uređaja s Home Assistantom razmotrit ćemo postupak integracije Samsung uređaja. Oni se moraju prvo povezati s vlastitom aplikacijom SmartThings [8]. Zatim, potrebno je unutar Home Assistanta pronaći istoimenu integraciju. Nakon toga, treba pratiti upute na ekranu za postavljanje uređaja. Ako sve završi bez problema, uređaj će biti vidljiv u web sučelju Home Assistanta i korisnik će moći upravljati njime.
Drugi primjer povezivanja je Aqara senzor prisutnosti što je zahtijevalo povezivanje na Aqara Home aplikaciju [9] prateći korake u uputstvima koja su priložena uz senzor. Unutar aplikacije provodi se konfiguracija samog senzora i tek nakon toga uređaj je vidljiv u Home Assistantu i može ga se integrirati pritiskom jednog gumba.

4.4. Home Assistant API

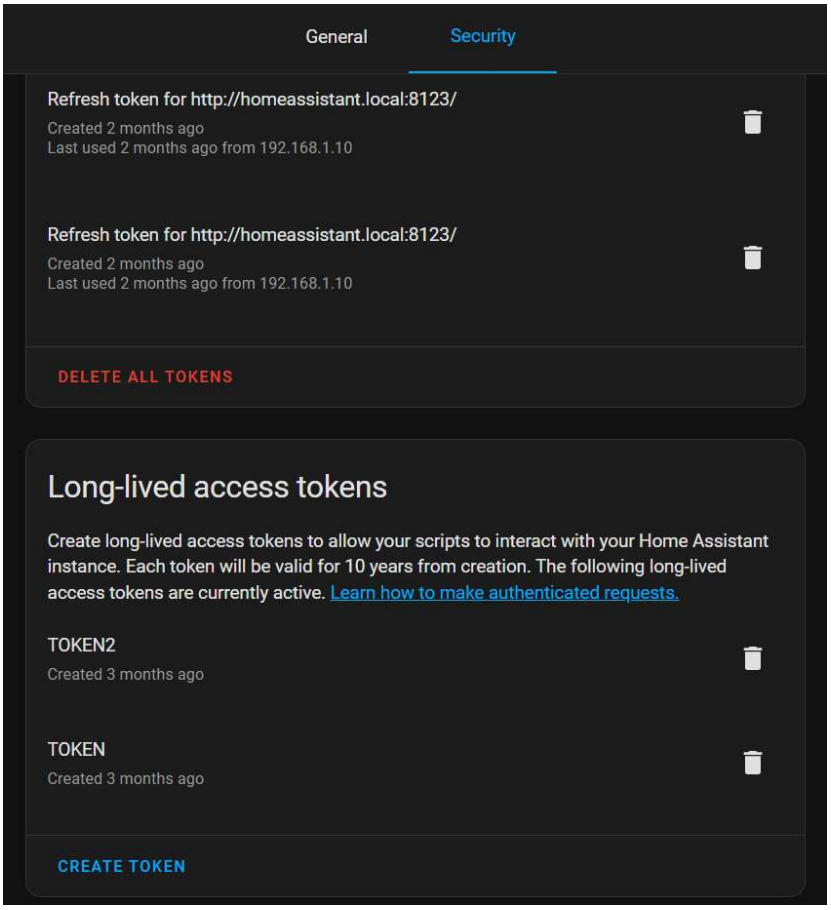
Home Assistant omogućuje upravljanje povezanim uređajima iz vlastitog programa koristeći njegov API. Kako bi zahtjevi koje se šalje bili autorizirani, potrebno je stvoriti vlastiti dugotrajni token (eng. Long-lived access token) koji se postavi u zaglavlje post

ili get zahtjeva (engl. request). Dugotrajni token je vrsta autentifikacijskog tokena koji omogućuje dugotrajno korištenje Home Assistant API-ja, odnosno nije potrebno kreirati novi token pri slanju svakog zahtjeva, a sigurnost podataka unutar zahtjeva je i dalje očuvana. Njega se kreira klikom na ikonu svog profila unutar Home Assistant web sučelja (Slika 4.1).



Slika 4.1: Bočni izbornik s prikazom ikone profila i korisničkog imena

Nakon toga otvara se stranica koja prikazuje vaše osobne podatke i dodatne mogućnosti. Klikom na karticu Security i pomicanjem stranice prema dolje (engl. scroll) prikazuje se gumb za generiranje dugotrajnog tokena (Slika 4.2).



Slika 4.2: Kartica za izradu tokena

Važno je odmah kopirati ovaj token jer, iz sigurnosnih razloga, nakon stvaranja neće biti moguće ponovno ga vidjeti u sučelju.

Nakon kreiranja tokena moguće je koristiti API pozive. Postoji nekolicina get i post zahtjeva čiji se opisi nalaze u njihovoj dokumentaciji. U svojoj implementaciji konkretno sam koristio dva. U sljedećim naredbama potrebno je zamijeniti `TOKEN` s vlastitim dugotrajnim tokenom kako biste autentificirali zahtjev.

4.4.1. GET /api/states

Za dohvaćanje stanja objekata koristi se sljedeća `curl` naredba:

```
curl \
  -H "Authorization: Bearer TOKEN" \
  -H "Content-Type: application/json" \
  http://homeassistant.local:8123/api/states
```

Odgovor (engl. response) na curl naredbu za ovu krajnju točku (engl. endpoint) izgleda ovako:

```
[
    {
        "attributes": {},
        "entity_id": "light.Ceiling",
        "last_changed": "2016-05-30T21:43:32.418+00:00",
        "state": "off"
    }
]
```

4.4.2. POST /api/services/<domain>/<service>

Pri korištenju ovog post zahtjeva moguće je poslati opcionalni JSON objekt koji će se koristiti kao `service_data`.

```
{
    "entity_id": "light.Ceiling"
}
```

Primjer `curl` naredbe za uključivanje svjetla:

```
curl \
  -H "Authorization: Bearer TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"entity_id": "entity_id": "light.Ceiling"}' \
  http://homeassistant.local:8123/api/services/homeassistant/turn_on
```

Vraća se popis stanja koja su se promijenila dok je usluga bila izvršavana.

```
[
    {
        "attributes": {},
        "entity_id": "light.Ceiling",
        "last_changed": "2016-05-30T21:43:32.418+00:00",
        "state": "on"
    }
]
```

5. Aplikacija

5.1. Korisnički zahtjevi

5.1.1. Funkcionalni zahtjevi

Funkcionalni zahtjevi opisuju očekivano ponašanje i mogućnosti koje aplikacija treba ponuditi korisniku. Aplikacija će razlikovati 2 vrste korisnika: javnog korisnika i prijavljenog korisnika. U nadolazećim potpoglavljima bit će opisati funkcionalni zahtjevi za navedene vrste korisnika.

Javni korisnik

Aplikacija je personalizirana za svakog korisnika stoga se javni korisnik mora prijaviti ili registrirati u sustav za daljnje korištenje aplikacije.

- Prijava u sustav
  - Korisnik se prijavljuje u sustav unosom korisničkog imena i lozinke. U slučaju neispravnog unosa nekog od podataka aplikacija će obavijestiti korisnika o tome, no radi pojačanja sigurnosti aplikacije ne smije odati koji je podatak neispravno unesen.
- Registracija u sustav
  - Korisnik se registrira u sustav unosom obaveznih podataka: korisničkog imena i lozinke. Aplikacija provjerava jedinstvenost korisničkog imena i je li ponovljena lozinka jednaka prethodno upisanoj te ako su ta dva zahtjeva ispunjena prebacuje korisnika na Login stranicu.

Prijavljeni korisnik

- Upravljanje LongLivedTokenima
  - Korisnik može dodati i brisati tokene za svoja kućanstva, tj. Home Assistant račune zato što je svaki račun vezan uz jednu lokaciju.

- Kreiranje optimizacije
    - Korisniku se prikazuju uređaji koji se nalaze u kućanstvu, tj. koji su spojeni na Home Assistant, odabranom u već spomenutom padajućem izborniku. Pored svakog uređaja nalazi se polje u koje korisnik može upisati broj sati koji želi da uređaj radi tog dana, a ako ne želi uključiti neki uređaj u optimizaciju ostavlja to polje prazno. Klikom na gumb korisnik poziva optimizacijski model sa zadanim ograničenjima koje je korisnik zadao te pokreće optimizaciju i automatsko upravljanje uređajima shodno izlazu optimizacijskog modela.
- Pregled i upravljanje svim kreiranim modelima
    - Korisnik može pregledati sve prethodno pokrenute automatizacije kreirane na temelju izlaza optimizacijskih modela, ponovno ih pokrenuti te ih izbrisati.
- Praćenje i nadjačavanje rada automatizacije
    - Korisnik tijekom automatskog upravljanja uređajima vidi prikazan graf s informacijama o uređajima kojima ona upravlja, tj. linijski graf koji prikazuje kada će uređaj raditi, a kada ne za svaki sat u danu, a ispod grafa će se nalaziti kartice s nazivima uređaja i kontrolnim gumbima za paljenje i gašenje s kojima će korisnik moći nadjačati automatsko upravljanje uređajima.

5.1.2. Nefunkcionalni zahtjevi

Nefunkcionalni zahtjevi definiraju tehničke aspekte aplikacije. Glavni zahtjevi su:

- Pristupačnost
    - Aplikacija mora ispravno raditi bez obzira na web preglednik koji korisnik koristi.
    - Aplikacija mora podržavati istovremeni rad više korisnika.
- Dostupnost
    - Aplikacija mora biti dostupna čak i u slučaju neispravnog korištenja.
- Performanse
    - Aplikacija mora izvršavati korisničke zahtjeve u razumnom vremenskom okviru.

- Sigurnost
    - Aplikacija mora omogućiti autentifikaciju korisnika i ograničiti funkcionalnosti prema korisničkim ulogama.
    - Aplikacija mora zaštititi osobne podatke korisnika, posebno lozinke.
    - Podaci moraju biti sigurno preneseni između klijenta i poslužitelja koristeći protokol poput HTTPS.
    - Aplikacija mora imati sigurnu vezu prema bazi podataka.
- Jednostavnost
    - Korisničko sučelje mora biti prilagodljivo na promjenu veličine prozora (engl. responsive) i intuitivno za sve korisnike aplikacije.

5.2. Tehnologije i alati

5.2.1. Tehnologije baze podataka

Baza podataka kreirana je u sustavu PostgreSQL [11], objektno-relacijskom sustavu za upravljanje bazama podataka otvorenog koda. PostgreSQL sadrži sve uobičajene značajke relacijskih baza podataka te podržava standardnu SQL sintaksu upita te pruža mehanizme za osiguravanje integriteta podataka, uključujući sigurnosne kopije, ograničenja integriteta transakcije i mehanizme oporavka podataka. Također, pruža visoke performanse i skalabilnost što ga čini izvrsnim izborom alata za pohranu korisničkih podataka u aplikaciji.
Pristup i upravljanje bazom podataka ostvareno je s pomoću alata pgAdmin, koji pruža grafičko sučelje za pregled relacija i ograničenja te pisanje i pregled rezultata SQL upita i slično.

5.2.2. Tehnologije poslužiteljske strane aplikacije

Za razvoj aplikacije korišten je programski jezik Python u kombinaciji s web okvirom Django [12]. Python je poznat po svojoj jednostavnosti i čitljivosti što omogućuje brzi razvoj i održavanje aplikacija. Django, koji je web okvir za Python, nudi bogat skup ugrađenih funkcionalnosti poput ORM-a (Object-Relational Mapping), autentifikacije korisnika, upravljanja sjednicama (engl. session) i administracijskog sučelja. Ova kombinacija omogućuje učinkovito i brzo razvijanje web aplikacija s minimalnim potrebama za dodatnim konfiguracijama, čineći je idealnom za razvoj ove aplikacije.

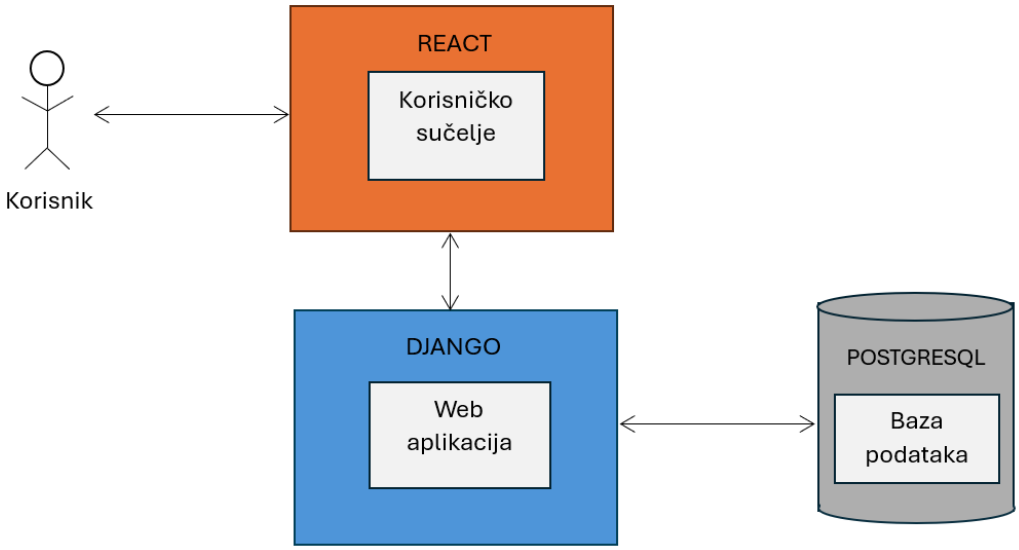
5.2.3. Tehnologije klijentske strane

Klijentska strana aplikacije izrađena je koristeći React [13], JavaScript biblioteku za izradu korisničkih sučelja. React omogućuje komponentni pristup razvoju, gdje se korisničko sučelje dijeli na male, ponovno iskoristive komponente. HTML i CSS se koriste za strukturu i stilizaciju stranica, dok se JavaScript koristi za dinamičku interakciju s korisnikom.

Za testiranje i debugiranje korisničkog sučelja te interakcije klijentske i poslužiteljske strane aplikacije korišten je Chrome web preglednik s ugrađenim alatom Chrome DevTools. Chrome DevTools omogućuje inspekciju elemenata, praćenje mrežnih zahtjeva, analizu performansi i debugiranje JavaScript koda, čineći ga neophodnim alatom u razvoju moderne web aplikacije.

5.3. Arhitektura sustava

Arhitektura sustava temelji se na kombinaciji Django (poslužiteljska strana), React (klijentska strana) i PostgreSQL (baza podataka). Ovaj pristup omogućuje jasnu podjelu odgovornosti unutar sustava, olakšava održavanje, testiranje i proširivost aplikacije. Način komunikacije korisnika, korisničkog sučelja, poslužitelja i baze podataka prikazan je na slici 5.1.



Slika 5.1: Komunikacija između korisnika, korisničkog sučelja, poslužitelja i baze podataka

5.3.1. Model baze podataka

U okviru ove aplikaciji bilo je potrebno spremati tri različita tipa podatka u bazu: podatke o korisniku (Tablica 5.1), njegove LongLivedTokene (Tablica 5.3) i optimizacijske modele koje je kreirao (Tablica 5.2). U nastavku su navedeni nazivi atributa, tip podataka te kratki opis uloge atributa za svaki entitet.

Tablica 5.1: Entitet "user"

Field Name	**Field Type**	**Description**
id	Integer (Primary Key)	Jedinstveni identifikator korisnika
username	CharField	Jedinstveno korisničko ime
password	CharField	Lozinka

Tablica 5.2: Entitet "longLivedToken"

Field Name	**Field Type**	**Description**
id	Integer (Primary Key)	Jedinstveni identifikator za svaki token
token	CharField	Token
homeAttached	CharField	Kućanstvo kojem token pripada
created_at	DateTimeField	Datum i vrijeme kreiranja tokena
author_id	Integer (ForeignKey)	Referenca na kreatora tokena

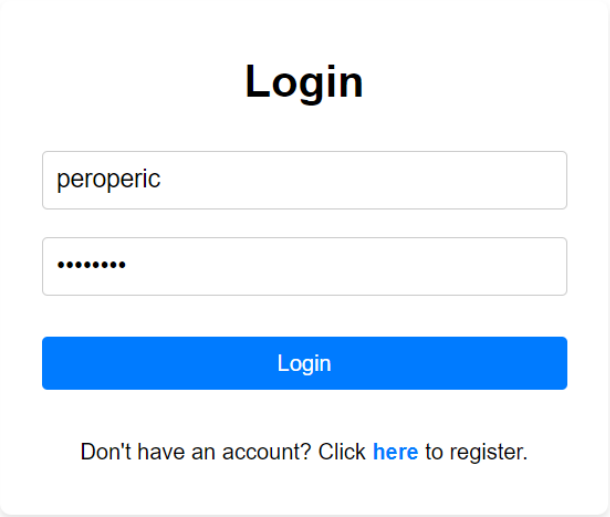
Tablica 5.3: Entitet "optimizationResult"

Field Name	**Field Type**	**Description**
id	Integer (Primary Key)	Jedinstveni identifikator kreiranog modela optimizacije
name	CharField	Ime modela
homeAttached	CharField	Kućanstvo za koje je model napravljen
homeToken	CharField	Token kućanstva nad kojim je izrađen optimizacijski model
result	JSONField	Rezultat optimizacijskog modela
created_at	DateTimeField	Datum i vrijeme kreiranja modela
author_id	Integer (Foreign Key)	Referenca na kreatora modela

5.4. Upute za korištenje aplikacije

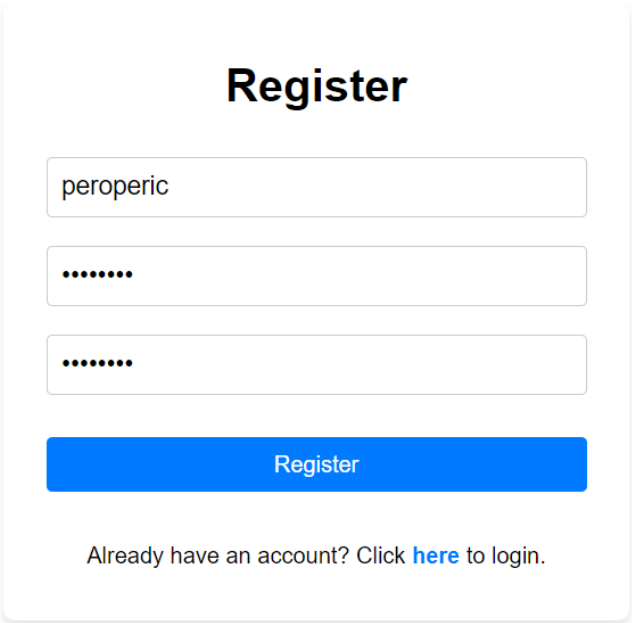
5.4.1. Prijava/registracija u sustav

Korisnik se mora prijaviti u sustav za daljnje korištenje. Prijavu izvršava upisivanjem korisničkog imena i lozinke na odgovarajuća mjesta u obrascu za prijavu (Slika 5.2) te klikom na gumb "Login".



Slika 5.2: Obrazac za prijavu u sustav

U slučaju da korisnik nema kreiran korisnički račun, klikom na poveznicu "here" otvara se obrazac za registraciju (Slika 5.3). Korisnik mora upisati jedinstveno korisničko ime i lozinku te ponovljenu lozinku.

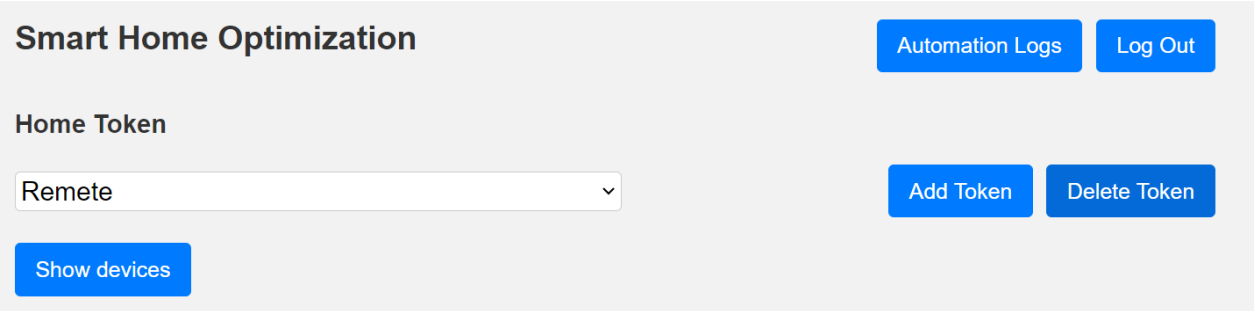


Slika 5.3: Obrazac za registraciju računa

Nakon toga, klikom na gumb "Register" korisnik je kreirao račun te se može pristupiti aplikaciji prijavom na obrazac Login.

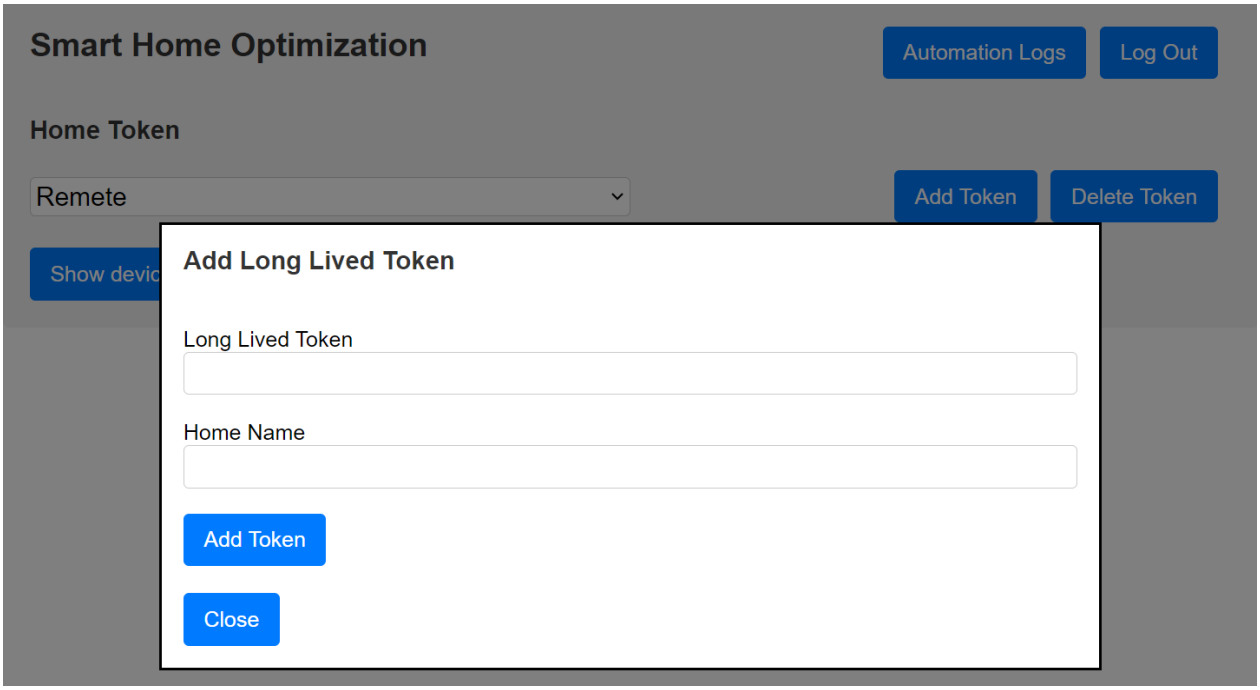
5.4.2. Dodavanje tokena

Na početnoj stranici aplikacije (Slika 5.4) nalazi se padajući izbornik s lijeve strane, a na desnoj nalaze se dva gumba.



Slika 5.4: Početna stranica

Klikom na gumb "Add Token" otvara se pomoćni prozor s obrascem za unos Long-lived access tokena i imena kućanstva na koje se gore uneseni token odnosi (Slika 5.5).

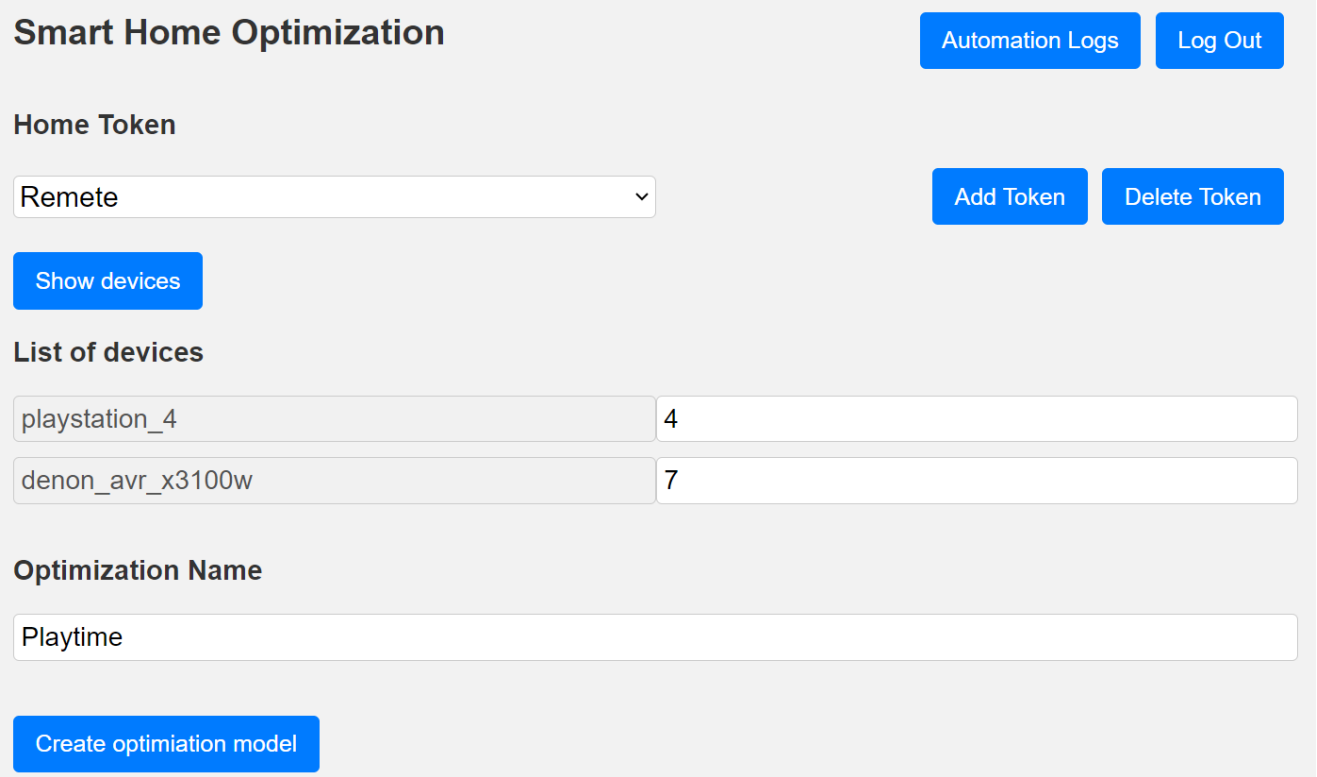


Slika 5.5: Obrazac za dodavanje tokena

Ponovnim klikom na gumb "Add Token" aplikacija obavještava korisnika o uspješnosti pohrane. Klikom na gumb "Delete Token" token se briše te aplikacija obavještava korisnika o uspješnosti akcije.

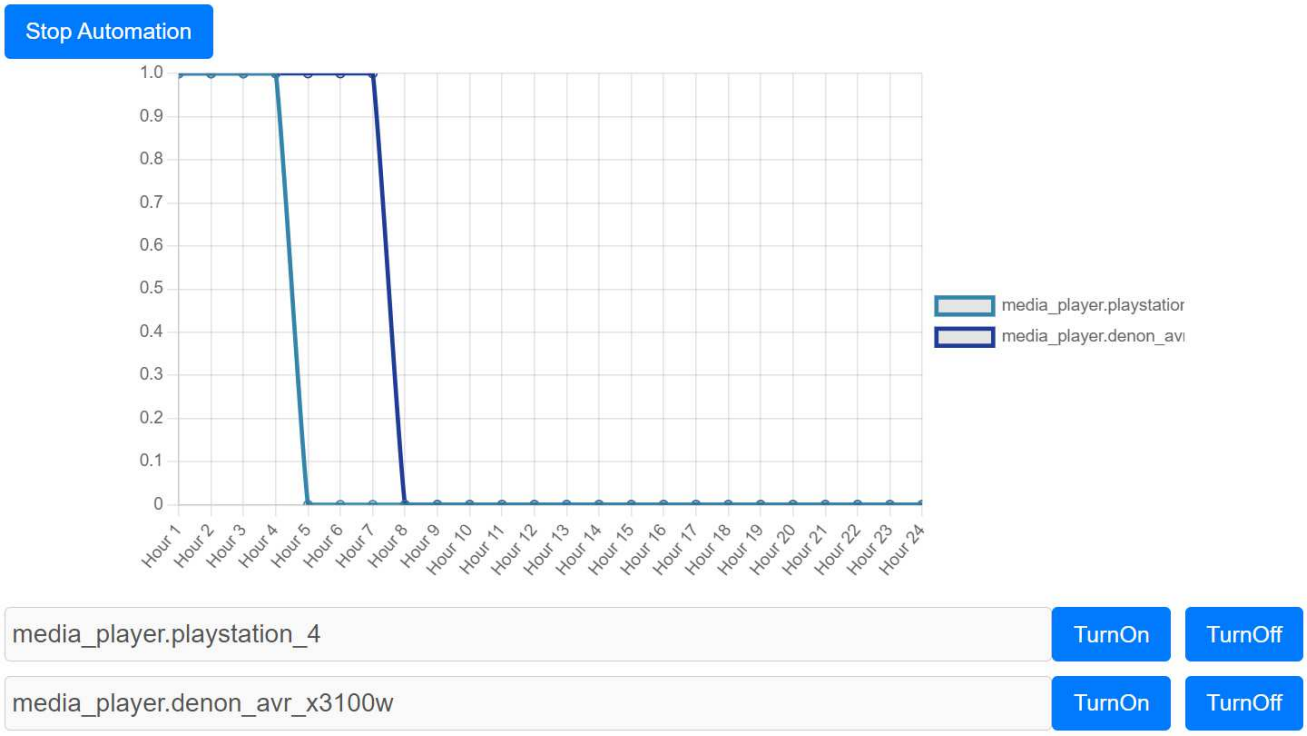
5.4.3. Kreiranje optimizacijskog modela i pokretanje automatizacije

Uz pretpostavku da je token koji je trenutno odabran u padajućem izborniku upravo token Home Assistanta koji se nalazi na istoj mreži kao i vaše računalo preko kojeg se pristupa ovoj aplikaciji, klikom na gumb "Show Devices" prikazat će se lista uređaja s poljem za unos sati koliko želimo da uređaj radi tijekom dana. Ako računalo nije spojeno na istu mrežu kao i Home Assistant ili trenutno nije odabran odgovarajući token, aplikacija obavještava korisnika o grešci. Ispod liste uređaja prikazuje se i polje za unos imena optimizacijskog modela kojeg izrađujemo (Slika 5.6) .



Slika 5.6: Prikaz uređaja spojenih na HA

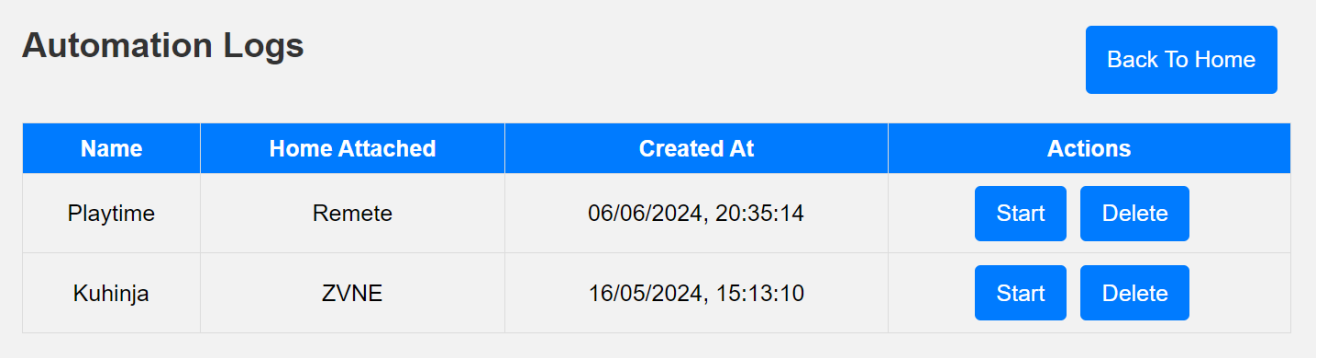
Klikom na gumb "Start optimization" kreira i pokreće se optimizacijski model (korištenjem koda iz poglavlja 3.1.2) s postavljenim parametrima za navedene uređaje, pokreće se automatsko upravljanje uređajima te se otvara vizualizacija automatizacije (Slika 5.7). Tako korisnik može pratiti stanje rada uređaja s obzirom na izlaz optimizacijskog modela.



Slika 5.7: Vizualizacija tijeka automatizacije

Također, ako korisnik želi nadjačati automatsku naredbu o paljenju ili gašenju određenog uređaja može to učiniti klikom na gumbe "TurnOn" i "TurnOff" koji će trenutno upaliti ili ugasiti uređaj. Graf će se shodno tome ažurirati. Klikom na pojedini uređaj u legendi na grafu, sakrit će liniju koja prikazuje stanje kliknutog uređaja. Ova funkcionalnost omogućuje jednostavnije praćenje potrošnje pojedinog uređaja u slučaju kada se upravlja većim brojem uređaja u kućanstvu. Klikom na gumb "Stop Automation" prekida se upravljanje uređajima i ponovno se otvara početna stranica.

Podaci o automatizacija koja se pokrenula na temelju optimizacijskog modela će se spremiti te će biti vidljivi na stranici "Automation Logs" kojoj se pristupa klikom na istoimeni gumb. Za svaku automatizaciju prikazano je ime optimizacijskog modela na temelju kojeg automatski upravlja uređajima, ime kućanstva u kojem je pokrenuta te datum i vrijeme stvaranja. Ondje je moguće pokrenuti neku od prije kreiranih automatizacija klikom na gumb "Start" te ih obrisati klikom na gumb "Delete" (Slika 5.8).



Slika 5.8: Popis automatizacija

5.4.4. Moguće poteškoće u radu s Home Assistantom i aplikaciji

Glavni problem u realizaciji ovog rada nastaje kada se svi uređaji pokušaju integrirati u jednu centralnu platformu, tj. Home Assistant.
Konstantna ažuriranja ovlaštenih aplikacija za povezivanje uređaja s Home Assistantom, kao što su npr. Smart Things za Samsung uređaje ili Aqara Smart Home za Aqara uređaje, i ažuriranja sustava Home Assistanta dovode nestabilnost u ovaj pristup. Pri izradi rada dogodila se potreba za ažuriranjem Home Assistant operacijskog sustava kao i sustava aplikacije Aqara, ali to nije bilo učinjeno odmah. Rezultat je bio greška u povezanosti i bilo je potrebno izvršiti ažuriranja na obje platforme kako bi Home Assistant mogao ponovno komunicirati s Aqara Presence senzorom.
Također, svakako treba osigurati stalnu povezanost svih uređaja kao i Home Assistanta s internetom iz istog razloga.

6. Zaključak

U sklopu ovog završnog rada proučen je princip odziva potrošnje i njegova implementacija u kućanstva. U vidu ostvarivanja tog cilja bilo je potrebno istražiti mogućnosti upravljanja pametnim uređajima, odnosno njihovom potrošnjom električne energije. Pametni uređaji su povezani na Home Assistant te je Home Assistant API omogućio upravljanje uređajima iz vlastitog programa.

Model baze podataka izrađen je u PostgreSQL-u, a web aplikacija implementirana je koristeći React na klijentskoj strani te radni okvir Django na poslužiteljskoj strani. Aplikacija omogućuje korisnicima određivanje broja sati rada u danu za pojedini uređaj na temelju čega se vrši optimizacija nad matematičkim modelom kojom se određuju periodi u kojima će kućanski uređaji raditi tako da se smanje troškovi električne energije, ali uz poštivanje ograničenja koja je korisnik zadao. Aplikacija će na temelju perioda rada automatski upravljati uređajima koristeći Home Assistant API pozive.

Sljedeći koraci u vidu proširenja rada bili bi implementacija strojnog učenja upotrebom nekih od algoritama poput linearne regresije, stabla odluke ili neuronskih mreža, koji bi uz podatke o potrošnji koje aplikacija prikuplja, mogli predvidjeti buduću potrošnju te upravljati uređajima na temelju automatski kreiranog optimizacijskog modela. Također, postavljanjem dodatnih senzora, koji mogu mjeriti količinu električne energije koju svaki uređaj koristi u jedinici vremena, i povezivanjem tih uređaja na Home Assistant bit će moguće prikazati točan trošak električne energije u svakom trenutku unutar aplikacije.

7. Literatura

[1] IEA 50 (2024.), Bazilian, M. and Van de Graaf, T., Pristupljeno 15.5.2024. (https://www.iea.org/)

[2] HEP ELEKTRA (2016.), HEP d.d., Pristupljeno 15.5.2024. (https://www.hep.hr/elektra/kucanstvo/tarifne-stavke-cijene/1547)

[3] Gurobi Optimization (2024.), Pristupljeno 15.5.2024. (https://www.gurobi.com/documentation/)

[4] Pandas (2024.), NumFOCUS, Inc., Pristupljeno 15.5.2024. (https://pandas.pydata.org/docs/)

[5] Home Assistant (2024.), Pristupljeno 5.5.2024. (https://www.home-assistant.io/)

[6] Hasanagić, S., Tehnologije internet stvari. Magistarski rad. Sveučilište Jurja Dobrile u Puli (2016.)

[7] Home Assistant Community (2024.), Pristupljeno 5.5.2024. (https://community.home-assistant.io/)

[8] Smart Things Developers (2024.), Samsung Electronics Co., LTD., Pristupljeno 5.5.2024. (https://developer.smartthings.com/docs/)

[9] Aqara (2023.), Lumi United Technology Co., LTD., Pristupljeno 5.5.2024. (https://www.aqara.com/us/support/user-manual/)

[10] Ljubas, D., Aplikacija za praćenje rekreativnih sportskih aktivnosti pojedinca. Završni rad. Fakultet elektrotehnike i računarstva u Zagrebu (2021.)

[11] PostgreSQL Documentation (2024.), The PostgreSQL Global Development Group, Pristupljeno 15.5.2024. (https://www.postgresql.org/docs/)

[12] Django documentation (2024.), Django Software Foundation, Pristupljeno 25.5.2024. (https://docs.djangoproject.com/en/5.0/)

[13] React (2024.), Meta Open Source, Pristupljeno 15.5.2024. (https://react.dev/)

[14] StackOverflow (2024.), Stack Exchange Inc., Pristupljeno 15.5.2024. (https://stackoverflow.com/)

[15] Malbašić, B., Određivanje fleksibilnosti uređaja u Laboratoriju za odziv potrošnje. Diplomski rad. Fakultet elektrotehnike i računarstva u Zagrebu (2022.)

Optimizacija rada kućanskih uređaja s obzirom na fluktuaciju cijena električne energije

Sažetak

U ovom radu opisan je proces proučavanja i izrade aplikacije u svrhu ostvarivanja optimizacije rada kućanskih uređaja s obzirom na fluktuacije cijena električne energije. U drugom poglavlju je objašnjen princip odziva na potrošnju. U trećem poglavlju opisano je stvaranje matematičkog modela optimizacije i implementacija modela u Pythonu. U četvrtom poglavlju opisani su načini instalacije Home Assistanta, povezivanje uređaja s Home Assistantom i kako se koristi Home Assistant API. U petom poglavlju govori se o funkcionalnim i nefunkcionalnim zahtjevima koje izrađena aplikacija treba imati. Također, opisane su tehnologije koje su se koristile za implementaciju te arhitektura aplikacije. Na kraju petog poglavlja dane su detaljne upute za korištenje izrađene aplikacije i opisani su problemi pri izradi aplikacije te spajanju uređaja s Home Assistantom.

Ključne riječi: odziv potrošnje, optimizacija, matematički model,Home Assistant, Python, Gurobi, API, React, Django, PostgreSQL, web aplikacija, automatizacija.

Optimization of Household Appliances Operation In Response To Electricity Price Fluctuations

Abstract

This paper describes the process of studying and developing an application aimed at optimizing the operation of household appliances in response to fluctuations in electricity prices. The second chapter explains demand response. The third chapter describes the creation of a mathematical optimization model and its implementation in Python. The fourth chapter outlines the installation methods of Home Assistant, connecting devices to Home Assistant, and how to use the Home Assistant API. The fifth chapter discusses the functional and non-functional requirements that the developed application needs to meet. Additionally, the technologies used for implementation and the architecture of the application are described. At the end of the fifth chapter, detailed instructions for using the created application are given and problems with creating the application and connecting devices with Home Assistant are described.

Keywords: demand response, optimization, mathematical model, Home Assistant, Python, Gurobi, API, React, Django, PostgreSQL, web application, automation.