

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1152

**WEB APLIKACIJA ZA PRIKAZ PODATAKA U OKRUŽJU  
PAMETNOG DOMA**

Ivan Lazarušić

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1152

**WEB APLIKACIJA ZA PRIKAZ PODATAKA U OKRUŽJU  
PAMETNOG DOMA**

Ivan Lazarušić

Zagreb, lipanj 2023.

Zagreb, 10. ožujka 2023.

## **ZAVRŠNI ZADATAK br. 1152**

Pristupnik: **Ivan Lazarušić (0036534657)**  
Studij: Elektrotehnika i informacijska tehnologija i Računarstvo  
Modul: Računarstvo  
Mentor: doc. dr. sc. Pavle Skočir

Zadatak: **Web aplikacija za prikaz podataka u okružju pametnog doma**

### Opis zadatka:

U današnjem Internetu raste broj jednostavnih uređaja poput senzorskih čvorova koji odašilju različite podatke prikupljene sa senzora. Da bi se od tih podataka dobile konkretne informacije, potrebno ih je pohraniti i pregledno prikazati korisnicima. Podaci se mogu pregledavati na različitim uređajima, televizorima, stolnim i prijenosnim računalima ili pokretnim uređajima poput tableta, pametnih telefona i pametnih satova. Vaš zadatak je koristiti programski alat Home Assistant na uređaju Raspberry Pi za prikupljanje senzorskih podataka i za upravljanje sustavom za osvjetljenje Philips Hue. Potrebno je razviti web aplikaciju koja se spaja na Home Assistant i omogućuje uvid u senzorske podatke i upravljanje uređajima spojenima na Home Assistant. Predloženo rješenje testirajte na odabranom studijskom slučaju.

Rok za predaju rada: 9. lipnja 2023.



## Sadržaj

Uvod .....	1
1. Internet stvari i pametni dom.....	2
1.1. Internet stvari.....	2
1.1.1. Arhitektura IoT-a.....	3
1.1.2. Komunikacija u IoT-u .....	4
1.1.3. Sigurnost u IoT-u.....	5
1.1.4. Primjena IoT-a.....	6
1.2. Pametni dom u sklopu interneta stvari .....	7
1.2.1. Osnovne karakteristike i prednosti .....	7
1.2.2. Izazovi .....	8
1.2.3. Primjeri primjene.....	8
2. Home Assistant i biblioteka React.....	11
2.1. Uvod u Home Assistant.....	11
2.2. Osnove REST API-ja .....	12
2.3. REST API i Home Assistant .....	12
2.4. Philips Hue sustav u sklopu Home Assistanta .....	13
2.4.1. Opis i funkcionalnost.....	13
2.4.2. Komponente sustava.....	14
2.4.3. Integracija Philips Hue-a i Home Assistanta.....	14
2.5. Biblioteka React .....	14
2.5.1. Osnovni koncepti biblioteke React.....	15
2.6. Integracija Reacta s Home Assistantom .....	16
2.6.1. Komunikacija s Home Assistant REST API-jem.....	16
2.6.2. Stanje komponenti i upravljanje Home Assistant entitetima.....	17
3. React web aplikacija za prikaz podataka .....	19

3.1.	Arhitektura sustava .....	19
3.2.	Konkretni primjeri komunikacije u kodu .....	20
3.3.	Opis sučelja konfiguracijske komponente .....	22
3.3.1.	Izvedba u kodu .....	23
3.4.	Opis sučelja nadzorne ploče .....	27
3.4.1.	Izvedba u kodu .....	27
3.5.	LinearGaugeChart komponenta.....	30
3.6.	GaugeChart komponenta .....	32
3.7.	Opis sučelja komponente prikaza povijesnih podataka .....	34
3.7.1.	Izvedba u kodu .....	35
3.8.	HistoryChart komponenta.....	36
3.9.	Opis ispitivanja studijskih slučajeva .....	38
	Zaključak .....	40
	Literatura .....	41
	Sažetak.....	43
	Summary.....	44

# Uvod

Pametni domovi postaju sve više zastupljeni u današnjem svijetu, pružajući napredne tehnološke mogućnosti i unapređujući udobnost i učinkovitost življenja. U sklopu ovog završnog rada, fokus je na razvoju web aplikacije koja omogućuje korisnicima pregled i upravljanje podacima u kontekstu pametnog doma. Ključni element u ovoj aplikaciji je Home Assistant, platforma otvorenog koda koja se koristi za automatizaciju različitih aspekata doma. Cilj je stvoriti preglednu i intuitivnu web aplikaciju koja će korisnicima pružiti uvid u senzorske podatke te omogućiti upravljanje uređajima povezanim s Home Assistant platformom.

Razvoj pametnih domova donosi brojne prednosti. Kroz integraciju različitih uređaja i senzora, korisnici mogu pratiti i kontrolirati razne aspekte svog doma, uključujući rasvjetu, temperaturu, sigurnost i energetske učinkovitost. Senzori prikupljaju podatke koji se zatim obrađuju i prikazuju korisnicima, pružajući im uvid u stanje doma i mogućnost da poduzmu odgovarajuće akcije. Uz to, povezanost s Home Assistant platformom omogućuje korisnicima da prilagode postavke i upravljaju uređajima preko web sučelja.

U radu će se detaljno razmotriti koncept pametnih domova, naglašavajući važnost integracije uređaja i senzora te mogućnost praćenja i upravljanja podacima u stvarnom vremenu. Također će se istražiti Home Assistant platforma kao ključni alat za automatizaciju i upravljanje pametnim domom. Prikazat će se implementacija web aplikacije koja će koristiti React biblioteku za stvaranje korisničkog sučelja i povezivanje s Home Assistant platformom. Kroz primjer studijskog slučaja, bit će demonstrirane praktične primjene i prednosti ove web aplikacije u kontekstu pametnog doma.

Razvoj web aplikacije za prikaz podataka u pametnom domu ima širok spektar potencijalnih primjena, pružajući korisnicima kontrolu, udobnost i efikasnost.

# 1. Internet stvari i pametni dom

U današnjem dobu, tehnološki napredak raste iz dana u dan velikom brzinom. Takav silovit napredak uvelike mijenja način na koji živimo, radimo i interagiramo s okolinom. Internet stvari (*engl. Internet of things*) i pametni domovi samo su jedna od mnogobrojnih ključnih područja koja su proizašla iz ovakvog rapidnog tehnološkog napretka.

Internet stvari predstavlja koncept koji se odnosi na povezivanje svakodnevnih objekata i uređaja s internetom zbog razmjene podataka i obavljanja raznih korisnih zadataka. Ovaj koncept omogućuje fizičkim objektima da dobiju pridjev “pametni”. To znači da takvi fizički objekti imaju mogućnost komunicirati s korisnicima i međusobno putem internetske mreže. [1]

S druge strane, pametni domovi su primjena interneta stvari u kontekstu stambenih prostora. Pametni domovi koriste tehnologije koje im je omogućio internet stvari kako bi omogućili automatizaciju, nadzor i kontrolu različitih aspekata kućnog okruženja. Takvi aspekti mogu uključivati temperaturu, sigurnost doma, kućanske aparate i brojne druge faktore povezane s udobnošću, sigurnošću i energetsom učinkovitošću. Pametni domovi koriste senzore, mreže i aplikacije za prikupljanje i analizu podataka, te omogućuju korisnicima upravljanje svim dijelovima njihovog doma na daljinu, prilagodbu postavki i stvaranje personaliziranih iskustava. U idućim poglavljima bit će detaljnije istraženi osnovni koncepti interneta stvari i pametnih domova.

## 1.1. Internet stvari

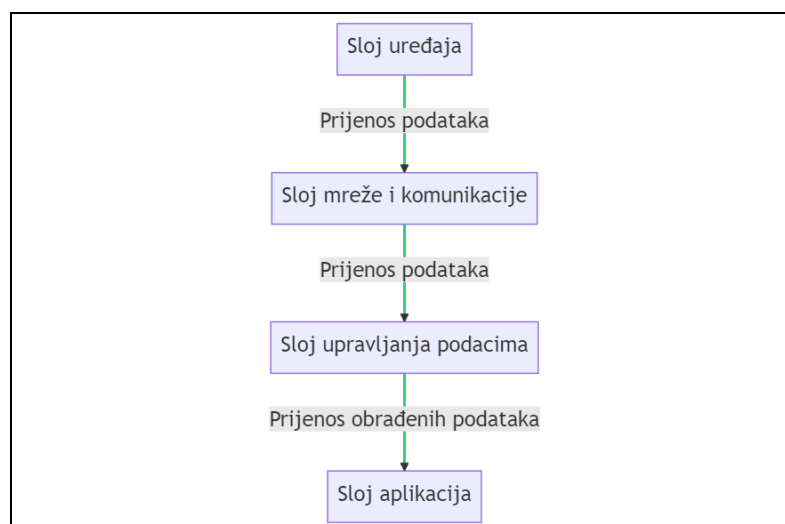
Internet stvari nastao je zbog brzog tehnološkog napretka i danas može biti dio gotovo svakog dijela našeg svakodnevnog života. U svojoj osnovi, internet stvari je mreža međusobno povezanih fizičkih uređaja, vozila, zgrada i drugih objekata opremljenih elektronikom, softverom, senzorima i mrežnim povezivostima. Takva mreža omogućuje njegovim korisnicima da automatiziraju podatke, upravljaju procesima, razmjenjuju podatke i komuniciraju s korisnicima.



### 1.1.1. Arhitektura IoT-a

Za arhitekturu interneta stvari možemo reći da je slojevita. Sastoji se od nekoliko slojeva. Brojni su pogledi na model arhitekture interneta stvari. Najrašireniji pogled na model je onaj sa četiri osnovna sloja (Slika 1.1) [2].

1. **Sloj uređaja** (*engl. Device layer*): Ovo je najniži sloj arhitekture i on se sastoji od samih fizičkih objekata – uređaja ili senzora. Uređaji mogu biti bilo što. Sve od pametnih telefona i automobila do perilica, termostata i kućne rasvjete.
2. **Sloj mreže i komunikacije** (*engl. Network and Communications Layer*): Ovaj sloj uključuje sve tehnologije koje su ključne u komunikaciji i povezivanju uređaja iz prijašnjeg sloja s internetom. Vrlo česte su tehnologije bežične komunikacije poput Wi-Fi, Bluetooth, LoRaWAN i druge.
3. **Sloj upravljanja podacima** (*engl. Data Management Layer*): Na ovom sloju se obavljaju zadatke vezane uz prikupljanje, pohranu i obradu podataka koje dolaze iz uređaja. Podaci koji se primaju od uređaja često su u nekim primitivnim formatima pa ovaj sloj također ima za zadaću pretvoriti podatke u neke korisnije formate.
4. **Sloj aplikacija** (*engl. Application Layer*): Zadatak ovog sloja je korištenje podataka iz prijašnjeg sloja za stvaranje korisničkih aplikacija. Aplikacije mogu biti bilo što od sustava za nadzor i kontrolu do inteligentnih sustava za analizu podataka koji korisnicima pružaju uvid u podatke i omogućuju im da donose informirane odluke.



Slika 1.1 Slojevi arhitekture IoT-a

### 1.1.2. Komunikacija u IoT-u

Bez efikasne komunikacije i pravilnog prijenosa podataka IoT tehnologije ne bi bilo moguće ostvariti. Uređaji i sustavi ne bi mogli razmjenjivati informacije ili međusobno djelovati na koordiniran način. Zato je važno proučiti neke od najosnovnijih aspekata komunikacije u internetu stvari:

1. **Protokoli komunikacije:** Protokoli su set pravila koji određuju na koji se način podaci primaju i prenose među uređajima. Postoje brojni protokoli, a u aplikacijama se koriste oni koji najbolje zadovoljavaju zahtjeve aplikacije kao što su efikasnost, domet, brzinu prijenosa podataka i sigurnost.

Najčešće korišteni protokoli su [3]:

- **Wi-Fi:** Ovaj protokol vjerojatno je najpopularniji protokol današnjice u okviru interneta stvari. Prednosti su mu dostupnost i velike brzine prijenosa podataka. Udaljenost na kojoj Wi-Fi nesmetano funkcionira varira između 50 i 100 metara.
- **Bluetooth:** Popularan protokol koji se primarno koristi za uređaje koji moraju komunicirati na malim udaljenostima.
- **Zigbee i Z-Wave:** Glavna im je upotreba u stvaranju mreža s malom potrošnjom energije u okruženju pametnog doma. Njihove karakteristike im omogućuju nesmetanu komunikaciju velikog broja malih uređaja.
- **LoraWAN (Slika 1.2):** Protokol koji se danas koristi za komunikaciju uređaja na velikim udaljenostima (2.5 km u urbanim područjima) s malom potrošnjom energije.

2. **Sigurnost u komunikaciji:** Kako broj uređaja u internetu stvari raste, tako rastu i zahtjevi sigurnosti u komunikaciji. Osnove sigurnosti komunikacije uključuju kriptiranje i autentifikacija komunikacije.
3. **Topologija mreže:** Topologija mreže opisuje način na koji su uređaji unutar mreže povezani. Neke od često korištenih topologija su zvjezdaste topologije (svi uređaji povezani s jednim središnjim čvorom), mrežne topologije (svaki uređaj može komunicirati sa svakim drugim uređajem, nema središnje točke), hijerarhijske topologije (svi uređaji su organizirani u različite razine) i topologije točka-na-točku (izravna povezanost dvaju uređaja). [4]



Slika 1.2 LoraWAN

### 1.1.3. Sigurnost u IoT-u

Uređaji u internetu stvari često prikupljaju osjetljive informacije koje su primamljive hakerima i cyber kriminalu. Kako te informacije ne bi dospjele u krive ruke, uređaji i njihova komunikacija trebaju biti adekvatno zaštićeni. Osnovne komponente sigurnosti unutar interneta stvari su [5] :

1. **Strategije zaštite:** Svaki dio IoT sustava mora zadovoljiti sigurnosne mjere. To uključuje:

- Sigurnost uređaja: Odnosi se na skup mjera koji moraju štiti sami uređaj od neovlaštenog pristupa. To uključuje korištenje zadovoljavajućih zaporki, ažuriranje firmwarea i softvera i sl.
- Sigurnost mreže: Mrežni kanali mogu biti podložni prisluškivanju ili korištenju za maliciozne napade, stoga trebaju postojati određene sigurnosne mjere koje će učiniti komunikaciju kroz mrežu sigurnom. To može uključivati autentifikaciju i autorizaciju uređaja, kriptiranje komunikacije i detekciju i spriječavanje napada na mreži.
- Sigurnost podataka: Ako se dogodi da sigurnosne mjere podbace i napadač uspije pribaviti podatke koje nije smio, treba se ostvariti da napadač iz tih podataka ne može napraviti nikakvu štetu. Sigurnosne mjere koje možemo primjeniti su kriptiranje podataka, pravilno upravljanje podacima i anonimizaciju podataka.

## 2. **Zakonodavni okvir:**

Osim što je sigurnost potrebno ostvariti zbog vlastitih i korisničkih interesa, nju je potrebno i ostvariti zbog različitih regulativnih zakona i propisa na različitim razinama – lokalno, nacionalno, međunarodno. To uključuje sigurnosne standarde, propise o zaštiti podataka i razne druge zakone koji se odnose na sigurnost u IoT.

### **1.1.4. Primjena IoT-a**

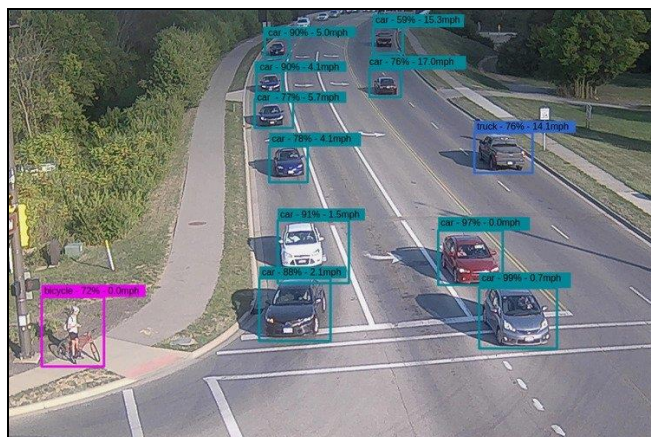
Tehnologije Interneta stvari danas su toliko raširene da se praktički svaki stanovnik urbanih područja jednom susreo s njima, bilo u poslovnom ili kućnom okruženju. Neka od najčešćih područja primjene su pametni domovi, zdravstvena zaštita, pametni gradovi i poljoprivreda.[6]

Pametni domovi su jedno od najčešćih područja primjene IoT-a. Razni uređaji u kućnom okruženju danas su modificirani tako da se mogu povezati na internet i dati korisnicima potpunu daljinsku kontrolu nad svojim domom. U idućim poglavljima značajke pametnih domova biti će detaljnije istraženi.

U zdravstvenoj zaštiti IoT omogućuje optimizaciju operacija u bolnicama, poboljšanje usluga skrbi i daljinsko praćenje pacijenata. IoT to omogućuje uz pametne narukvice ili satove koji prate vitalne znakove pacijenata, sustave za upravljanje bolničkim inventarom, te pametne pilule koje precizno doziraju lijekove pacijentima.

Pametni gradovi koriste IoT za poboljšanje gradske infrastrukture i kvalitete života građana. Primjene uključuju upravljanje otpadom, praćenje kvalitete zraka, upravljanje parkiralištima, upravljanje prometnom rasvjetom (Slika 1.3) i drugo.

U poljoprivredi IoT tehnologije mogu donijeti značajne novitete. Kroz upotrebu senzora i naprednih analitičkih alata, poljoprivrednici lakše mogu pratiti stanje svojih usjeva i stoke, poboljšati upravljanje resursima kao što su zemlja i gnojivo, te se lakše prilagoditi vremenskim uvjetima.



Slika 1.3 Sustav pametnog semafora

## 1.2. Pametni dom u sklopu interneta stvari

Prepoznatljiv trend u današnjem tehnološkom svijetu postali su pametni domovi. Oni predstavljaju jedan od temeljnih dijelova IoT-a. Inovacije u okviru pametnih domova pormijenile su način na koji razmišljamo o udobnosti, energetskej učinkovitosti, pa čak i zabavi u našim privatnim prostorima. U idućim potpoglavljima biti će detaljnije istražene temeljen značajke pametnih domova.

### 1.2.1. Osnovne karakteristike i prednosti

Osnovne karakteristike koje zadovoljava praktički svako okruženje pametnog doma uključuju:

1. **Automatizacija:** Stvari koje su u prošlosti bila kućanska briga ukućana danas se može svesti samo na dva klika na pametnom telefonu. Na primjer, moguće je postaviti da se svjetla u pametnom domu pale u točno određeno vrijeme, ili je moguće postaviti klima uređaj da se automatski uključi kada temperature prijeđe određenu granicu.
2. **Učinkovitost energije:** Automatizacijom možemo utjecati i na učinkovitije upravljanje potrošnjom energije u domovima. Na primjer, upravljanje temperaturom u domu može biti toliko precizno da se ne troši ni trunca više energije nego što je potrebno. Takvo učinkovito upravljanje energijom ima pozitivnije financijske, a i ekološke učinke.

3. **Daljinska kontrola:** Pametni domovi omogućavaju svojim korisnicima da upravljaju svim aspektima svoga doma bez obzira koliko daleko se korisnici nalazili od svojih domova.
4. **Personalizacija:** Uporaba tehnologija pametnog doma omogućuju svojim korisnicima da personaliziraju svoje domove prema svojim potrebama i preferencijama. Na primjer, korisnik može isprogramirati rasvjetu, glazbu ili temperaturu točno prema svojem ukusu.

### 1.2.2. Izazovi

Iako je očigledno kako pametni domovi imaju brojne prednosti, u pojedinim situacijama stvaraju i određene izazove na koje treba obratiti pozornost. Neki od takvih izazova su sigurnost podataka, kompatibilnost i interoperabilnost, te složenost postavljanja i upravljanja.

Unatoč mnogih sigurnosnih mjera koje se provode pri dizajniranju IoT kućnih uređaja, treba se uzeti u obzir da se praktički svaki takav uređaj mora spojiti na internet kako bi nesmetano funkcionirao. Samo spajanje uređaja na internet nosi rizik od neovlaštenog pristupa i zloupotrebe podataka. Napadači stalno pronalaze nove načine kako probiti i najčvršće sigurnosne mjere, pa se konstantno treba ulagati u što bolju sigurnost.

Običnim korisnicima koji nisu detaljno upućeni u današnji robusni tehnološki svijet nekada nije lako razlučiti kako integrirati uređaje različitih proizvođača da oni funkcioniraju optimalno. Pametni domovi mogu pravilno funkcionirati jedino ako su integracija i interoperabilnost među njihovim uređajima savršeno zadovoljeni. Manje upućenim korisnicima također je nekada upravljanje i postavljanje pametnih domova dosta kompliciran proces. Naime, potrebno je proučiti i razumjeti različite uređaje, protokole i platforme kako bi se postigla optimalna funkcionalnost, što nije tako jednostavno.

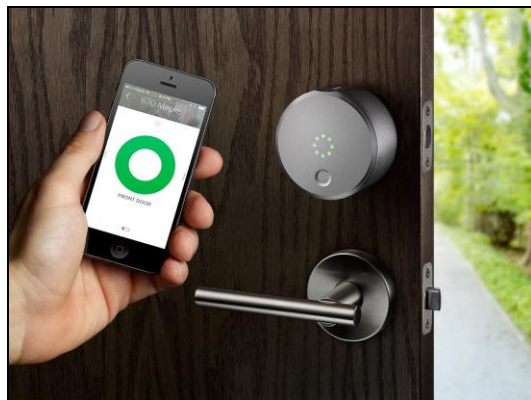
### 1.2.3. Primjeri primjene

Primjena pametnih domova obuhvaća različite aspekte domaćinstva i svakodnevnog života. Evo nekoliko konkretnijih primjera primjene pametnih domova [7]:

- Pametni termostati: Pametni termostati, poput Nest Learning Thermostata ili Ecobee termostata, omogućuju korisnicima da precizno kontroliraju temperaturu u svojim domovima. Korisnici mogu postaviti rasporede grijanja i

hlađenja prema svojim preferencijama te prilagoditi temperaturu putem pametnih uređaja, čime se postiže energetska učinkovitost i udobnost.

- Pametne brave: Pametne brave, kao što su August Smart Lock (Slika 1.4) ili Yale Assure Lock, omogućuju korisnicima da zaključavaju i otključavaju vrata putem pametnih uređaja. To omogućuje udobnost daljinskog pristupa domu, dijeljenje virtualnih ključeva s obitelji i gostima te praćenje aktivnosti ulaska i izlaska



Slika 1.4 August Smart Lock

- Pametne video-nadzorne kamere: Pametne video-nadzorne kamere, kao što su Ring Doorbell ili Arlo Pro, omogućuju korisnicima da nadziru svoje domove putem video snimaka visoke razlučivosti. Korisnici mogu pratiti svoje dvorište, ulazna vrata ili unutarnje prostorije putem pametnih uređaja, primiti obavijesti o detekciji pokreta i čak komunicirati s posjetiteljima putem dvosmjerne komunikacije.
- Pametni sustavi zaštite od požara i poplava: Pametni sustavi zaštite od požara i poplava, poput Nest Protect i Honeywell Lyric Water Leak and Freeze Detector, omogućuju korisnicima da budu obaviješteni o potencijalnim opasnostima u njihovom domu. Ovi uređaji detektiraju dim, visoku temperaturu ili prisutnost vode te šalju obavijesti korisnicima putem pametnih uređaja, omogućujući brzu reakciju i zaštitu doma.
- Pametni sustavi zaštite od provala: Pametni sustavi zaštite od provala, poput ADT Pulse i SimpliSafe, omogućuju korisnicima da nadziru sigurnost svog

doma putem senzora pokreta, magnetskih kontakata na vratima i prozorima te kamera. Korisnici mogu primiti obavijesti o potencijalnom pokušaju provaljivanja, daljinski upravljati alarmnim sustavom i pratiti video snimke.

- Pametni kućanski aparati: Mnogi kućanski aparati su dostupni u pametnim verzijama, kao što su pametne perilice i sušilice rublja, hladnjaci s mogućnošću upravljanja putem pametnih uređaja ili pametni kuhalo za vodu. Ovi uređaji omogućuju korisnicima daljinsku kontrolu, praćenje i optimizaciju rada kućanskih aparata.



## 2. Home Assistant i biblioteka React

Početak drugog poglavlja uključuje dublje istraživanje Home Assistanta i biblioteke React, dva ključna alata za stvaranje sofisticiranih rješenja za automatizaciju pametne kuće. Iako ova dva alata na prvi pogled mogu izgledati nevezano, njihova kombinacija otvara širok spektar mogućnosti za korisnike, omogućavajući im da kreiraju prilagođene integracije i sučelja koja su prilagođena njihovim specifičnim potrebama.

Home Assistant predstavlja središnji dio ovog poglavlja, pružajući infrastrukturu koja omogućava korisnicima kontrolu nad mnoštvom IoT uređaja unutar njihovih domova. Sa fokusom na zaštitu privatnosti i lokalnu kontrolu, Home Assistant predstavlja idealnu platformu za one koji žele zadržati punu kontrolu nad svojim uređajima.

S druge strane, biblioteka React predstavlja moćan alat za izgradnju interaktivnih korisničkih sučelja. Njegova modularna priroda omogućava korisnicima da kreiraju sofisticirane aplikacije koje su lako održavati i proširivati.

Međutim, prava snaga ovih alata dolazi do izražaja kada se koriste zajedno. Kroz integraciju Reacta s Home Assistantom, korisnici mogu kreirati prilagođene interakcije i automatizacije, pružajući sebi i drugima intuitivna sučelja za upravljanje svojim pametnim kućama.

### 2.1. Uvod u Home Assistant

**Home Assistant** je platforma za automatizaciju kućnih uređaja koja se fokusira na zaštitu privatnosti i lokalnu kontrolu. Ova open source platforma nudi mogućnost integracije i upravljanja mnoštvom različitih IoT uređaja, od svjetiljki do klima uređaja. Cilj platforme je omogućiti korisnicima da ostvare "pametnu kuću" bez potrebe za upotrebom oblaka, osiguravajući pri tome privatnost i sigurnost podataka.

Home Assistant ima snažno sučelje koje omogućava korisnicima da programiraju složene automatizacije, scenarije i rutine. Načini integracije uređaja uključuju lokalne mreže, Bluetooth, Zigbee, Z-Wave, MQTT protokol i, naravno, HTTP REST API.

## 2.2. Osnove REST API-ja

**REST** (Representational State Transfer) je softverski arhitekturni stil koji definira set pravila za izradu web usluga. Jedna od ključnih značajki REST-a je da koristi standardne HTTP metode, što ga čini jednostavnim za razumijevanje i korištenje. Ove metode uključuju GET (dohvati podatke), POST (stvari podatak), PUT (ažuriraj podatak) i DELETE (izbriši podatak).

**REST API** (Application Programming Interface) omogućava komunikaciju između različitih softvera preko HTTP protokola. API-ji su sučelja koja omogućavaju softverima da komuniciraju jedan s drugim bez potrebe za korisničkom interakcijom. U kontekstu Home Assistanta, REST API omogućava drugim programima da šalju zahtjeve za kontrolu uređaja ili dohvaćanje statusa uređaja.

## 2.3. REST API i Home Assistant

Home Assistant nudi **RESTful API**, što znači da se koristi REST arhitektura za interakciju s uređajima. Ovo sučelje omogućava korisnicima da upravljaju svojim uređajima preko **HTTP zahtjeva**.

Da bi se ostvarila veza između Home Assistanta i uređaja, prvo je potrebno konfigurirati Home Assistant tako da zna kako komunicirati s uređajem. To uključuje postavljanje IP adrese uređaja, porta na kojem uređaj sluša, kao i eventualne potrebne autentifikacijske podatke.

Kada je veza uspostavljena, Home Assistant može poslati HTTP zahtjev uređaju kako bi izvršio neku akciju. Na primjer, mogao bi poslati GET zahtjev da dohvati trenutno stanje senzora, ili POST zahtjev da promijeni stanje svjetla.

S druge strane, uređaj može odgovoriti na zahtjev potvrdom da je akcija izvršena ili pružanjem traženih podataka. Home Assistant zatim može iskoristiti ove informacije za ažuriranje sučelja ili pokretanje drugih akcija.

**REST API** je posebno koristan za uređaje koji ne podržavaju druge metode integracije s Home Assistantom, ili za situacije kada korisnik želi postići finu kontrolu nad uređajem. Ova interakcija je također vrlo korisna za razvijanje vlastitih prilagođenih integracija ili za integraciju uređaja koji nisu izravno podržani od strane Home Assistanta.

Zaključno, REST API i Home Assistant zajedno omogućuju širok raspon mogućnosti za kontrolu i automatizaciju pametnih uređaja u kući. Korisnicima omogućuju prilagođavanje svojih pametnih kuća prema vlastitim potrebama, uz poštivanje njihove privatnosti i sigurnosti.

## 2.4. Philips Hue sustav u sklopu Home Assistanta

### 2.4.1. Opis i funkcionalnost

Philips Hue je popularan sustav za pametno osvjtljenje koji omogućuje korisnicima upravljanje rasvjetom putem pametnih uređaja. Sustav se temelji na bežičnoj komunikaciji i koristi Zigbee protokol za komunikaciju između pametnih žarulja, mosta i upravljačkih uređaja.

Philips Hue žarulje (Slika 2.1) dolaze s različitim značajkama, uključujući mogućnost podešavanja svjetlosti, boje i efekata, kao i daljinsko upravljanje putem mobilnih aplikacija i glasovnih asistenata. Funkcionalnosti Philips Hue sustava uključuju:

- Promjena boje i intenziteta svjetlosti žarulja.
- Stvaranje prilagođenih svjetlosnih scena i efekata.
- Upravljanje osvjtljenjem putem mobilnih aplikacija i daljinskih upravljača.
- Automatizacija radnji temeljenih na vremenskim rasporedima i senzorskim podacima.



Slika 2.1 Philips Hue žarulje

## 2.4.2. Komponente sustava

Philips Hue sustav sastoji se od sljedećih ključnih komponenti:

- a. Philips Hue žarulje: To su pametne žarulje koje podržavaju bežičnu komunikaciju i omogućuju podešavanje boje i svjetlosti.
- b. Philips Hue Bridge (most): To je centralna upravljačka jedinica koja povezuje Philips Hue žarulje s ostalim uređajima i omogućuje njihovo upravljanje putem aplikacija i vanjskih integracija.
- c. Upravljački uređaji: Philips Hue sustav podržava različite načine upravljanja, uključujući mobilne aplikacije, daljinske upravljače, glasovne asistente i integraciju s drugim pametnim sustavima.

## 2.4.3. Integracija Philips Hue-a i Home Assistanta

Integracija između Home Assistanta i Philips Hue sustava omogućuje korisnicima centralizirano upravljanje i nadzor nad osvjetljenjem unutar pametnog doma. Home Assistant pruža mogućnost povezivanja s Philips Hue Bridge-om te dobivanja pristupa i kontrole nad Philips Hue žaruljama i funkcionalnostima.

Kako bi se ostvarila integracija, korisnik treba konfigurirati Philips Hue komponentu unutar Home Assistanta, što uključuje pružanje pristupnih podataka poput IP adrese mosta i pristupnog tokena. Nakon uspješne konfiguracije, Home Assistant stvara entitete za Philips Hue žarulje i omogućuje korisnicima upravljanje osvjetljenjem putem sučelja ili automatizacijom.

Integracija omogućuje različite scenarije i automatizacije, poput uključivanja i isključivanja svjetla na temelju prisutnosti senzora ili vremenskih rasporeda, podešavanja boje i svjetline svjetla ovisno o dobu dana ili prilagođenim korisničkim zahtjevima.

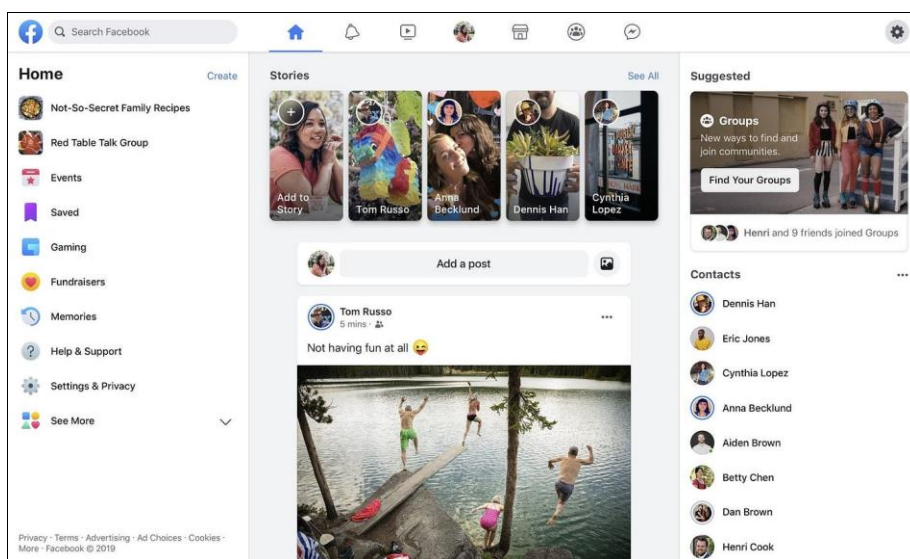
## 2.5. Biblioteka React

React je popularna JavaScript biblioteka koja se koristi za izgradnju korisničkih sučelja za web aplikacije. Razvijen od strane Facebooka, React se temelji na konceptu komponenti, što omogućava modularnost i ponovno iskoristivost koda. Glavni cilj Reacta je pružiti efikasno i deklarativno programiranje, kao i bolje upravljanje stanjem korisničkog sučelja.

## 2.5.1. Osnovni koncepti biblioteke React

React se sastoji od koncepata koji su bili revolucionarni u vrijeme kada je React izašao za javnu upotrebu. Neki od njih uključuju stvari kao što su [8]:

- **Komponente:** React aplikacije se grade od komponenti. Komponente su samostalni dijelovi korisničkog sučelja koji se mogu ponovno koristiti. Svaka komponenta ima svoj skup svojstava (props) i stanje (state). Cijela Facebook web aplikacija temeljena je na komponentama koje se lako mogu uočiti u njihovom vizualnom prikazu na web stranici (Slika 2.2)
- **Virtualni DOM:** React koristi virtualni DOM (Document Object Model) koji je interna reprezentacija stvarnog DOM-a. Virtualni DOM omogućava efikasno ažuriranje samo promijenjenih dijelova korisničkog sučelja umjesto cijelog stabla DOM elemenata.
- **JSX:** JSX (JavaScript XML) je proširenje sintakse JavaScripta koje omogućava pisanje HTML sličnog koda unutar JavaScripta. JSX olakšava izradu korisničkog sučelja u Reactu.
- **Komponentni životni ciklus:** Svaka React komponenta prolazi kroz različite faze svog životnog ciklusa, kao što su inicijalizacija, ažuriranje i uništavanje. Ove faze omogućavaju programerima da obavljaju određene akcije u određenim trenucima tijekom rada komponente.



Slika 2.2 Facebook UI temeljen na Reactu

## 2.6. Integracija Reacta s Home Assistantom

Home Assistant ima vlastiti frontend koji je izgrađen s pomoću JavaScripta i Reacta. Ovaj frontend je odgovoran za prikaz korisničkog sučelja, interakciju s korisnicima i upravljanje sustavom.

Integracija Reacta u Home Assistant omogućuje razvoj prilagođenih komponenti i sučelja. Kroz definiranje novih React komponenti može se dodati nove funkcionalnosti, vizualne elemente i interakcije s korisničkim sučeljem.

### 2.6.1. Komunikacija s Home Assistant REST API-jem

Da bi React aplikacija komunicirala s Home Assistantom, može se koristiti REST API koji pruža Home Assistant. REST API omogućuje izvođenje različitih akcija, kao što su dohvaćanje podataka o entitetima, upravljanje uređajima i slanje događaja.

React komponente mogu koristiti AJAX zahtjeve kako bi komunicirale s REST API-jem Home Assistanta i dohvaćale podatke ili izvršavale akcije. Na primjer, možete poslati zahtjev za dohvat trenutnog stanja senzora ili promijeniti postavke svjetla. Home Assistant ima u svojoj dokumentaciji veliki broj predefiniranih lokacija s kojima se može komunicirati pomoću REST API-ja. Neke od najčešće korištenih akcija navedene su u tablici ispod teksta (Tablica 2.1).

AKCIJA	FUNKCIONALNOST
GET /api/states	Vrati podatke o stanju pojedinačnih komponenti spojenih na Home Assistant npr. vrijednosti temperature, tlaka ili vlažnosti s obzirom o kakvoj komponenti se radi
POST /api/states/<entity_id>	Promijeni ili stvori novo stanje. Može se stvoriti bilo kakvo stanje bez obzira na to koji je entity_id predan kao krajnja lokacija

	(npr. Ovo koristimo ako hoćemo promijeniti svjetlinu pametne rasvjete)
GET /api/history/period/<timestamp>	Akcija s kojom možemo dobiti podatke o tome kako su se mijenjala pojedina stanja u nekoj komponenti s obzirom na period koji zatražimo s GET.

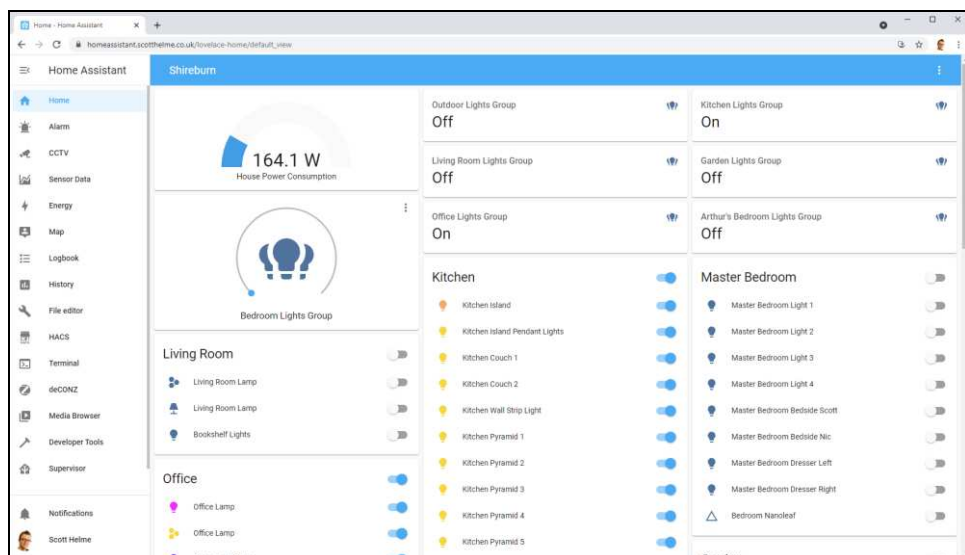
Tablica 2.1

Korištene akcije u projektnom zadatku

### 2.6.2. Stanje komponenti i upravljanje Home Assistant entitetima

Jedna od ključnih prednosti Reacta je upravljanje stanjem komponenti. React komponente mogu pratiti svoje vlastito stanje i ažurirati korisničko sučelje na temelju tih promjena.

Kroz REST API Home Assistanta, React komponente mogu pratiti i upravljati stanjem entiteta u Home Assistantu. Na primjer, može se ažurirati stanje svjetla kroz React komponentu, a promjena će se prikazati na korisničkom sučelju Home Assistanta (Slika 2.3). To je izrazito moćan alat u slučaju izrade vlastitih dinamičkih sučelja koristeći komunikaciju sa Home Assistantom. Reactova stanja komponenti omogućavaju i prikazivanje senzorskih podataka uživo, osvježavajući određene komponente koje su zadužene za to. Više o tome kako je takva funkcionalnost implementirana u ovom radu bit će u poglavljima koja se bave samom implementacijom programske potpore u sklopu projektnog završnog zadatka.



Slika 2.3

Jedan od mogućih izgleda Home Assistant korisničkog sučelja

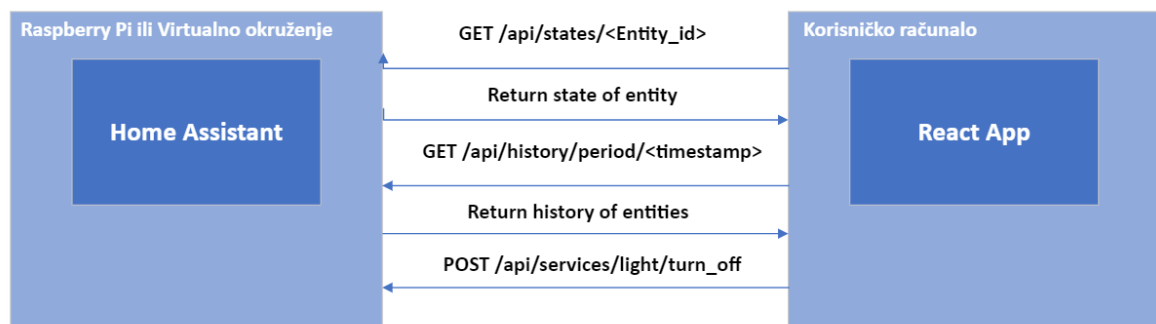


### 3. React web aplikacija za prikaz podataka

U ovom poglavlju se istražuje praktična programska implementacija web aplikacije koja koristi tehnologije opisane u poglavlju 2, kako bi izradili web aplikaciju sa sučeljem koje uživo prikazuje podatke dobivene od strane Home Assistanta i omogućuje upravljanjem komponenti koje su spojene na Home Assistant.

#### 3.1. Arhitektura sustava

U poglavlju 2.6.1 spomenuti su procesi spajanja React aplikacije i Home Assistanta. Najbolji način za to je korištenjem REST API-ja koji je srž arhitekture ovog programskog sustava. Na slici (Slika 3.1) prikazan je primjer konkretne komunikacije u sustavu koristeći REST API.



Slika 3.1

Komunikacija u sustavu

Prvi GET zahtjev služi React aplikaciji da dohvati informacije o trenutnim stanjima senzora ili svjetiljki. Te informacije aplikacija koristi kako bi vizualizirala podatke tog senzora ili svjetiljke na određenom grafikonu u komponenti nadzorne ploče.

Drugi GET zahtjev služi React aplikaciji da dohvati informacije o povijesnim podacima pojedinog entiteta. Umjesto “<timestamp>” može se napisati konkretnu vremensku oznaku koja označava od kojeg mjesta u vremena želimo dohvaćati povijesne podatke. Ove informacije aplikacija koristi kako bi prikazala povijesne podatke entiteta na vremenskim grafovima.

POST zahtjev ključan je za kontrolu statusa uključenosti pojedinog entiteta unutar Home Assistanta. Koristan je u komponenti nadzorne ploče u kojoj je implementiran prekidač koji kontrolira uključenost pametnog svjetla.

## 3.2. Konkretni primjeri komunikacije u kodu

REST zahtjevi koriste se u tri konkretne React komponente. To su nadzorna ploča, konfiguracijska komponenta i komponenta za prikaz povijesnih podataka.

U nadzornoj ploči unutar `useEffect` React kuke imamo sljedeći kod za dohvaćanje stanja:

```
const response = await axios.get(
  `http://${state.data}/api/states/${entityId.id}`,
  {
    headers: {
      Authorization: `Bearer ${state.bearer}`,
    },
  },
);
```

Kod 3.1

U React kodu se koristi paket Axios kako bi se uspješno obradio odgovor Home Assistanta. Ovaj kod hvata trenutno stanje (vrijednost) entiteta sa identifikatorom `entityId.id` na IP adresi `state.data`. Autorizacija zahtjeva obavlja se uspomoć vrijednosti tokena u varijabli `state.bearer` u zaglavlju zahtjeva. Konfiguracijska komponenta (početna stranica) predaje nadzornoj ploči identifikator, IP adresu i token kako bi ona mogla uspješno dohvaćati podatke o trenutnim stanjima entiteta uspomoć `useNavigation()` kuke s kojom šalje `state` varijablu nadzornoj ploči.

Unutar komponente za prikaz podataka na vremenskom grafu imamo drugi GET zahtjev (Kod 3.2). On je ključan za dohvaćanje povijesnih podataka u određenom vremenskom periodu za neki entitet. Za razliku od prijašnjeg GET zahtjeva, ovaj zahtjev ima i svoje parametre koje šalje skupa sa zaglavljem.

`filter_entity_id` služi za određivanje svih entiteta čije se povijesne podatke želi dobiti u odgovoru.

`end_time` služi za definiranje posljednje točke u vremenu do koje želimo primati podatke. `start.toISOString()` predstavlja početnu točku u vremenu od koje želimo primati povijesne podatke.

```
const response = await axios.get(
  `http://${ipAddress}/api/history/period/${start.toISOString()}
`,
  {
    headers: {
      Authorization: `Bearer ${bearer}`,
    },
    params: {
      filter_entity_id: entityId,
      end_time: end.toISOString(),
    },
  }
);
```

Kod 3.2

U nadzornoj ploči javlja se i POST zahtjev za kontroliranje statusa uključenosti pametnog svjetla (Kod 3.3)

```
await axios.post(
  `http://${state.data}/api/services/light/${
    lightState ? "turn_off" : "turn_on"
  }`,
  {
    entity_id: "light.virtual_light",
  },
  {
    headers: {
      Authorization: `Bearer ${state.bearer}`,
    },
  }
);
setLightState(!lightState); // Update local state
```

Kod 3.3

`lightState` je stanje ostvareno uspomoć Reactove kuke za stanja na idući način:

```
const [lightState, setLightState] = useState(false);
```

To stanje se mijenja u ovisnosti sa stanjem prekidača na nadzornoj ploči. Ako je stanje `false`, šalje se POST zahtjev sa `turn_off` kako bi se svjetiljka u Home Assistantu ugasila. Slično funkcionira i za stanje `true`, samo se svjetiljka onda pali.

### 3.3. Opis sučelja konfiguracijske komponente

Konfiguracijska komponenta početna je stranica web aplikacije. Njen osnovni zadatak je omogućiti korisniku unos IP adrese, tokena za autentifikaciju te postavljanje i dodavanje svih informacija vezanih uz senzore i svjetla. Sve te informacije se dalje proslijeđuju idućoj komponenti na daljnju obradu.

Na slici (Slika 3.2) vidljivi je izgled i dizajn konfiguracijske komponente web aplikacije.

U prvo polje za unos teksta unosi se IP adresa na kojoj je pokrenuta Home Assistant instanca. U konkretnom slučaju to može biti IP adresa Raspberry Pi uređaja na kojem je pokrenut Home Assistant ili pak IP adresa lokalnog računala (engl. *localhost*) na kojem je Home Assistant pokrenut unutar Docker kontejnera. Drugo polje služi za unos tokena za autentifikaciju POST i GET zahtjeva iz vanjskih izvora. Taj token se treba generirati i kopirati iz korisničkih postavki u Home Assistantu.

Nakon unosa IP adrese i tokena za autentifikaciju potrebno je dodati proizvoljan broj senzora ili svjetala za obradu i prikaz na idućim komponentama web stranice. Svakom senzoru ili svjetlu mora se definirati njegov identifikator entiteta, ime, mjerna jedinica podataka koje koristi i soba u kojoj se taj senzor nalazi. Nužno je da korisnik unese točan ID entiteta i mjernu jedinicu jer su oni ključni za točnu obradu podataka tog entiteta. ID pojedinačnog senzora ili svjetla može se pronaći u instanci Home Assistanta pod oznakom za razvojne alate (engl. *Developer tools*). Ime i sobu korisnik definira po vlastitim preferencijama.

Gumb “Submit” šalje sve unesene podatke idućoj komponenti. “Save configurations” sprema trenutačno unesene podatke u poljima na stranici i sprema ih u JSON file. “Load configurations” učitava podatke i automatski popunjava sva polja s predefiniranim podacima. Ovo omogućuje korisniku da koristi svoje preferirane postavke kada god to želi, bez da ih svaki put mora iznova unositi.

## Configuration Page

IP Address

localhost:8123


Long-Lived Access Token

.....


### Sensors & lights

<div>Entity id</div> <div>sensor.virtual_tem</div>	<div>Name</div> <div>Temperature</div>	<div>Unit</div> <div>°C</div>	<div>Room</div> <div>Dnevna soba</div>	⊖
<div>Entity id</div> <div>sensor.virtual_pres</div>	<div>Name</div> <div>Pressure</div>	<div>Unit</div> <div>hPa</div>	<div>Room</div> <div>Kuhinja</div>	⊖
<div>Entity id</div> <div>input_number.virtu</div>	<div>Name</div> <div>Brightness</div>	<div>Unit</div> <div>%</div>	<div>Room</div> <div>Kuhinja</div>	⊖
<div>Entity id</div> <div>sensor.virtual_humr</div>	<div>Name</div> <div>Humidity</div>	<div>Unit</div> <div>lx</div>	<div>Room</div> <div>Vrt</div>	⊖

⊕

 SUBMIT

LOAD CONFIGURATIONS

 SAVE CONFIGURATIONS

Slika 3.2 konfiguracijska komponenta

### 3.3.1. Izvedba u kodu

U ovom dijelu bit će razmatrani ključni dijelovi koda koji se odnose na izgradnju konfiguracijske stranice. Jedna od ključnih biblioteka koja se koristi zove se Material-UI.

Material-UI je popularna biblioteka za izradu sučelja u React.js, koja dolazi s predefiniranim komponentama koje pružaju brz i efikasan način za izgradnju atraktivnih i funkcionalnih korisničkih sučelja. Ovaj set komponenti slijedi princip Material dizajna koji je razvila Googleova dizajn grupa, a obuhvaća širok spektar kontrola, složenih sekcija i ikona.

U Kod 3.4, nakon uvoza korištenih biblioteka i ostalih zavisnosti, definiramo React funkcionalnu komponentu `ConfigurationPage`. Uz pomoć `useState` kuke kreiramo state varijable (npr. `ipAddress`, `password`) za spremanje unosa korisnika. Osim

varijabli navedenih u Kod 3.4 također se još definiraju state varijable za `names`, `units`, `rooms` koje služe za spremanje ostalih unosa korisnika, te `ipError`, `passwordError`, `sensorError` koje su inicijalno postavljene na `false` i prate je li neki od tih unosa korisnika zaboravio unijeti (ti unosi ne smiju biti prazni).

```
const ConfigurationPage = () => {
  const classes = useStyles();
  const navigate = useNavigate();
  const [ipAddress, setIpAddress] = useState("");
  const [password, setPassword] = useState("");
  //...
```

#### Kod 3.4 postavljanje stanja

Funkcija `validateForm` (Kod 3.4) provjerava jesu li uneseni podaci ispravni. Ako neki podaci nedostaju, funkcija postavlja odgovarajuće state varijable za grešku (npr. `setPasswordError`) na `true`.

```
const handleSubmit = (event) => {
  event.preventDefault();
  if (!validateForm()) {
    return;
  }
  ...
  navigate('/dashboard', { state: { data: ipAddress, array:
sensorData, bearer: password } });
  ...
};
```

#### Kod 3.5 Podnošenje obrasca

U Kod 3.5 `handleSubmit` funkcija upravlja podnošenjem obrasca. Ona najprije provjerava valjanost obrasca pomoću `validateForm` funkcije. Ako je obrazac valjan, korisnik se preusmjerava na nadzornu ploču (`' /dashboard'`) s unesenim podacima.

`sensorData` je niz koji se sastoji od jedinki koje predstavljaju senzore. Te jedinice dobivaju podatke o identifikatoru senzora, imenu senzora, mjernoj jedinici i imenu sobe u

kojoj se nalaze. U Kod 3.6 se može vidjeti punjenje niza `sensorData` sa takvim jedinkama u `handleSubmit` funkciji s obzirom na unose korisnika.

```
const sensorData = sensors.map((sensor, index) => ({
  id: sensor,
  name: names[index],
  unit: units[index],
  roomName: rooms[index] || "Not specified"
}));
```

Kod 3.6 Punjenje podataka o senzorima

U `return` dijelu koda ove komponente (Kod 3.7 Osnovni izgled `return`) možemo vidjeti kod koji se koristi za izgradnju korisničkog sučelja uspomoć brojnih Material-UI komponenti. Cijeli sadržaj stranice je enkapsuliran u `Card` komponenti

```
return (
  <Card className={classes.card}>
    ...
    <CardContent>
      <Typography variant="h4" component="h2"
className={classes.title} gutterBottom>
        Configuration Page
      </Typography>
      <form onSubmit={handleSubmit}>
        <Grid container spacing={2}
className={classes.grid}>
          ...
        </Grid>
      </form>
    </CardContent>
    ...
  </Card>
);
```

Kod 3.7 Osnovni izgled `return` dijela koda

Neke od korištenih MUI komponenata su:

- **Card:** Komponenta `Card` u `Material-UI` predstavlja površinu na kojoj su informacije prikazane u obliku povezanih blokova, koje obično predstavljaju neki oblik unosa ili informacija. To su fleksibilni i ekstenzibilni sadržajni spremnici koji sadrže različite elemente poput tekstova, slika, akcija i gumba.
- **CardContent:** Ova komponenta se koristi unutar `Card` komponente i služi kao spremnik za primarni sadržaj te kartice. Ona osigurava odgovarajući razmak unutar kartice za bolju vidljivost i čitljivost sadržaja.
- **Typography:** Komponenta `Typography` u `Material-UI` se koristi za prikaz teksta u UI. Ona podržava različite stilove i varijacije teksta, uključujući različite težine fonta, veličine i boje. Koristeći ovu komponentu, možemo osigurati dosljednost teksta diljem cijele aplikacije.
- **Grid:** `Grid` je komponenta koja koristi `CSS Flexbox` model za izgradnju strukture stranice ili komponente. To je moćan alat za izgradnju responzivnih layouta, što omogućava jednostavno upravljanje rasporedom komponenti na različitim veličinama zaslona.

Osim u ovoj komponenti za konfiguraciju, ove `Material-UI` komponente se koriste i kroz sve ostale dijelove aplikacije jednako koliko i na ovoj. Polja za korisnički unos ostvarena su uspomoc `TextField` i `Select` komponenti.

Stilovi se koriste uz pomoć `makeStyles` funkcije iz `@mui/styles` paketa. Ova funkcija koristi *CSS-in-JS* pristup za stiliziranje komponenti. Pomoću nje možemo kreirati stilove u JavaScriptu i kasnije ih primijeniti na `React` komponente.

```
const useStyles = makeStyles((theme) => ({
  card: {
    maxWidth: 800,
    margin: "auto",
    marginTop: "5%",
    padding: "24px",
    borderWidth: "2px",
  },
  title: {
    marginBottom: "16px", }, }));
```

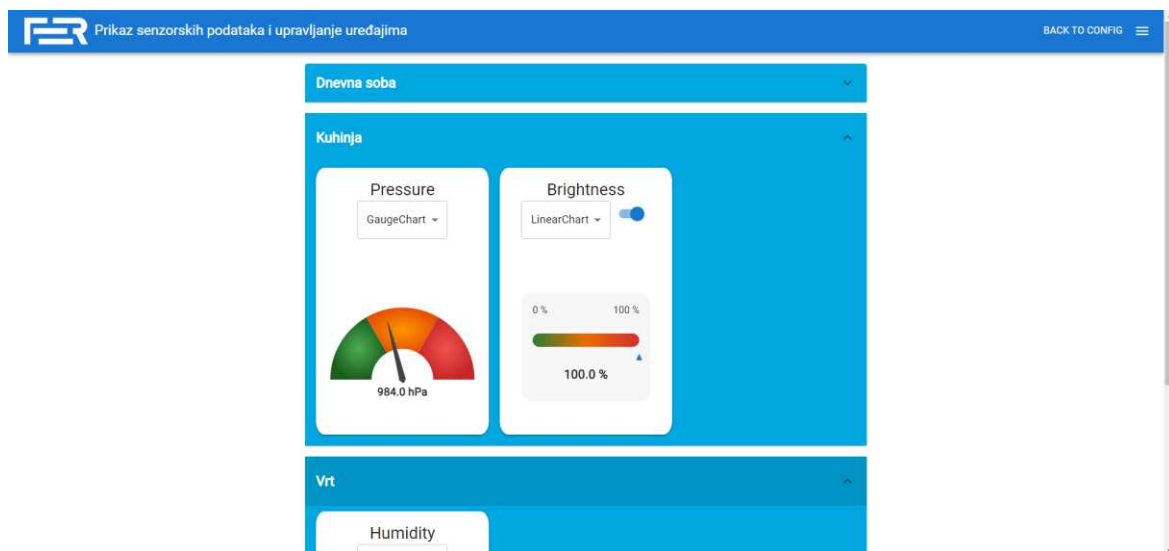
Kod 3.8 Jedan dio `makeStyles` funkcije



### 3.4. Opis sučelja nadzorne ploče

Sučelje nadzorne ploče ili *Dashboard* (Slika 3.3 Nadzorna ploča) sastoji se od zaglavlja i tijela stranice. U zaglavlju se nalaze informacije o trenutnoj komponenti, logo fakulteta, gumb za povratak na konfiguracijsku komponentu i padajući izbornik koji nudi mogućnost prebacivanja na komponentu prikaza povijesnih podataka.

U tijelu aplikacije vidljivi su linearni i cirkularni grafikoni senzorskih podataka podijeljeni po sobama koje je korisnik odlučio postaviti na konfiguracijskom stranici. Korisnik ima mogućnost da podatke svakog unesenog senzora prikazuje ili na linearnom ili na cirkularnom grafikonu. Posebni slučaj je prikaz grafikona za svjetlinu pametnog svjetla jer on ima i prekidač koji kontrolira status uključenosti pametnog svjetla.



Slika 3.3 Nadzorna ploča

#### 3.4.1. Izvedba u kodu

Početni dio izvedbe uključuje vidimo importiranje različitih modula i komponenti koje su neophodne za funkcioniranje Dashboard komponente. Nakon toga na sličan način kao u Kod 3.4 postavljanje stanjapostavljaju se početna stanja komponente koristeći `useState` kuke.

```
const [sensorsData, setSensorsData] = useState([]);
const [chartTypes, setChartTypes] = useState({});
const [lightState, setLightState] = useState(false);
```

### Kod 3.9 Postavljanje početnih stanja

U `sensorData` se učitavaju podaci koje je korisnik unio na konfiguracijskoj stranici. `chartTypes` se koristi za uspješnu promjenu i pamćenje trenutnog grafikona (linearni ili cirkularni) koji prikazuje podatke za određeni senzor. `lightState` je namijenjen za praćenje promjena prekidača uključenosti pametne svjetiljke. Inicijalno je postavljen na `false` jer je pretpostavka da je pri konfiguraciji senzor uključen, pa je i pritom i ugašenost svjetla postavljena na `false`.

`UseLocation()` kuka se koristi za dohvaćanje podataka sa konfiguracijske stranice i destrukciju podataka poslanih sa `useNavigate` u objektu `state`:

```
const { state } = useLocation();
```

Dashboard komponenta također implementira i dva GET zahtjeva Home Assistantu i jedan POST zahtjev koji služe za prikazivanje podataka senzora i gašenje odnosno paljenje pametnog svijetla. Funkcionalnosti i programski kod ovih zahtjeva detaljno su objašnjeni su u poglavlju 3.2.

U kodu je vidljiva funkcija `sensorsByRoom` koja je ključna za točno renderiranje stranice jer je ona zadužena za grupiranje podataka o senzorima prema sobama:

```
const sensorsByRoom = sensorData.reduce((result, sensor) =>
{
  result[sensor.roomName] = [...(result[sensor.roomName] ||
[]), sensor];
  return result;
}, {});
```

Važan dio koda odnosi se i na samu upotrebu `sensorsByRoom` funkcije. Ona se koristi u `return` naredbi komponente `Dashboard`.

```
{Object.entries(sensorsByRoom).map(([roomName, sensors]) =>
  (/* ...ostatak koda za renderiranje... */)
```

Koristeći metodu `map()`, prolazi se kroz ovaj niz parova ključ-vrijednost i za svaku sobu renderira odgovarajući JSX element. Unutar petlje, može se pristupiti sobi (`roomName`) i nizu senzora (`sensors`) za tu sobu i koristiti ih za točan prikaz po sobama.

```

return (
  <ThemeProvider theme={theme}>
    <Container maxWidth="xl">
      <Box sx={{ flexGrow: 1 }}>
        <Grid container spacing={2}>
          ...
          <Grid item xs={12}>
            {chartType === "GaugeChart" ? (
              <GaugeChart sensor={selectedSensor} />
            ) : (
              <LinearGauge sensor={selectedSensor} />
            )}
          </Grid>
        </Grid>
      </Box>
    </Container>
  </ThemeProvider>
);

```

Kod 3.10 sažeti oblik return dijela koda

Osim dijela koji filtrira senzore po sobama, možemo se dotaknuti i `return` dijela koda.

Kao i u komponenti za konfiguraciju koriste se Material-UI komponente. Neke koje nisu bile prije spomenute, a pojavljuju se u ovoj komponenti su:

- **ThemeProvider:** Komponenta koja omogućuje primjenu teme na aplikaciju. Tema definira stilizaciju komponenti, poput boja, fontova i razmaka.
- **Box:** Komponenta koja omogućuje stvaranje prilagođenih kontejnera i omotavanje elemenata s dodatnim stilovima. Ovdje se koristi `sx={{ flexGrow: 1 }}` za postizanje rastezljivosti i punog prostora unutar kontejnera.
- **Container:** Komponenta koja pruža kontejner s ograničenom širinom i centriranim sadržajem. Ovdje se koristi `maxWidth="xl"` za postizanje širine kontejnera na ekstra velikim zaslonima.
- **Accordion:** Komponenta koja omogućuje stvaranje sklopivih/sekcionálnih sadržaja. Prikazuje zaglavlje koje se može proširiti ili skupiti kako bi se otkrio ili

sakrio povezani sadržaj. Koristi se `AccordionSummary` komponenta kao zaglavlje i `AccordionDetails` komponenta za prikaz sadržaja kad je `Accordion` otvoren. U ovom slučaju, unutar `Accordiona` su smješteni grafikoni po sobama.

`LinearChart` i `GaugeChart` su samoizrađene React komponente za renderiranje grafova. Više o njima bit će rečeno u idućim poglavljima.

### 3.5. LinearGaugeChart komponenta

`LinearGaugeChart` je komponenta koja koristi koncept linijskog mjernog instrumenta za prikaz podataka. Omogućuje vizualizaciju vrijednosti na linijskoj skali s jasno označenim granicama.

```
const ColorLinearProgress = styled(LinearProgress) ({ theme,
value, min, low, high }) => {
  ...
});
```

Kod 3.11

Kod 3.11 definira `ColorLinearProgress`, što je varijacija na MUI komponentu `LinearProgress`. Ona implementira brojne stilove, no najbitniji definirani stil joj je:

```
backgroundImage: `linear-gradient(to right,
${theme.palette.success.main}, ${theme.palette.warning.main},
${theme.palette.error.main})`
```

koji definira da `ColorLinearProgress` ima linearni gradijent udesno, od zelene prema crvenoj, kako bi uspješno dočarala promjenu vrijednosti koja joj je poslana.

```
const ValueIndicator = styled('div') ({ theme, value }) => ({
  ...
});
```

Kod 3.12 ValueIndicator

`ValueIndicator` (Kod 3.12 ValueIndicator) služi kao pokazivač trenutne vrijednosti na mjerilu. Mjesto ovog pokazivača se dinamički mijenja ovisno o postotku vrijednosti. U `ValueIndicator` komponenti, `left` je CSS svojstvo koje određuje horizontalni

položaj elementa na stranici. Ovdje je korišteno za pozicioniranje indikatora vrijednosti na pravo mjesto na linijskom mjerilu:

```
left: `calc(${value}% - 5px)`
```

Vrijednost `left` je dinamički izračunata pomoću CSS funkcije `calc()`. Ova funkcija omogućuje izvođenje aritmetičkih operacija na CSS vrijednostima. U osnovi, indikator je pomaknut za `value` postotaka od lijevog ruba mjerila, a zatim se dodatno pomakne unazad za 5 piksela da bi bio centriran.

```
const LinearGaugeChart = ({ title, value, min, max, low,
  high, unit }) => {
  ...
};
```

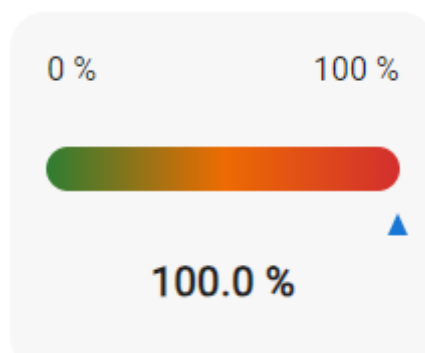
### Kod 3.13 LinearGaugeChart

Ova funkcija je glavna komponenta `LinearGaugeChart`. Koristi `ColorLinearProgress` i `ValueIndicator` za prikaz vrijednosti na mjerilu. Postotak napretka se izračunava na osnovu minimalne (`min`) i maksimalne (`max`) vrijednosti:

```
const progressValue = ((value - min) / (max - min)) * 100;
```

`value` predstavlja trenutnu vrijednost koju se želi prikazati, `min` i `max` su granične vrijednosti mjerila. Izraz `(value - min) / (max - min)` daje relativnu poziciju trenutne vrijednosti između minimalne i maksimalne vrijednosti. Množenje sa 100 pretvara ovaj broj u postotak.

Nakraju se taj `progressValue` šalje komponenti `ColorLinearProgress` na obradu i prikazivanje.



Slika 3.4 Izgled LinearChart komponente

## 3.6. GaugeChart komponenta

GaugeChart je komponenta koja koristi koncept kružnog mjernog instrumenta (ili brzinometra) za prikaz podataka. Slično LinearGaugeChartu, vizualizira vrijednost unutar određenog raspona, ali ovaj put na kružnom skali. Ostvaren je uspomoć ReactSpeedometer komponente iz paketa *react-d3-speedometer*. Umjesto ValueIndicator komponente, u GaugeChart komponenti kao pokazivač koristi se paketno ugrađena igla ReactSpeedometer komponente (*engl. needle*).

```
<ReactSpeedometer
  maxValue={max}
  minValue={min}
  value={value}
  needleColor={theme.palette.grey[800]}
  needleHeightRatio={0.7}
  needleBaseColor="none"
  segments={3}
  segmentColors=[
    radialGradient('green'),
    radialGradient('yellow'),
    radialGradient('red'),
  ]
  textColor={theme.palette.text.primary}
  needleTransitionDuration={2000}
  needleTransition="easeElastic"
  width={250}
  height={150}
  customSegmentLabels={createEmptyLabels(3)}
  currentValueText={` ${value} ${unit}`}
/>
```

Slika 3.5 ReactSpeedometer komponenta

Parametri `maxValue` i `minValue` određuju granice vrijednosti na brzinometru. To su gornja i donja granica kružnog mjerila. U konkretnom primjeru, `min` i `max` vrijednosti prenose se kroz propse, omogućavajući dinamičko postavljanje granica.

Parametar `value` predstavlja trenutnu vrijednost koja se želi prikazati na brzinometru. Ona se vizualno predstavlja pozicijom igle na mjerilu. Kako `value` varira, tako će se igla pomicati po skali.

Boja igle definira se kroz `needleColor`. Ova igla je ključan vizualni element jer pokazuje trenutnu vrijednost na skali.

Visina igle određena je kroz `needleHeightRatio`, a ovo je proporcija koja određuje visinu igle u odnosu na ukupnu visinu brzinometra. Na primjer, vrijednost 0.7 znači da će visina igle biti 70% visine brzinometra.

`segments` određuje broj segmenata na brzinometru. U ovom slučaju, definirana su 3 segmenta koji odgovaraju različitim područjima - zeleno, narančasto i crveno.

`segmentColors` određuje boje segmenata. Ovaj parametar prima niz boja, gdje svaka boja odgovara jednom segmentu. Ovdje se koristi funkcija `radialGradient` za stvaranje radialnog gradijenta koji će se koristiti za bojanje svakog segmenta.

`currentValueText` je tekst koji prikazuje trenutnu vrijednost na brzinometru. U ovom slučaju, prikazuje se trenutna vrijednost i jedinica mjere.

`needleTransitionDuration` i `needleTransition` su parametri koji kontroliraju animaciju igle. Ovi parametri definiraju koliko će dugo trajati tranzicija igle (u milisekundama) i koji će se efekt koristiti za tranziciju. U ovom primjeru, koristi se elastična tranzicija (`easeElastic`), što znači da će igla imati efekt odskoka kad se pomiče.

Ostali dijelovi `GaugeChart` komponente su više-manje orijentirani na stil i brušenje izgleda.



Slika 3.6 Izgled `GaugeChart` komponente

### 3.7. Opis sučelja komponente prikaza povijesnih podataka

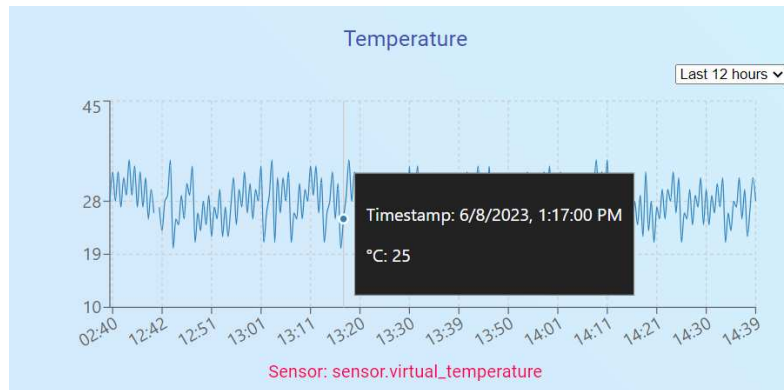
Na Sliku 3.7 Sučelje za prikaz povijesnih podataka može se vidjeti izgled sučelja za prikaz povijesnih podataka (*History Panel*). Izgled ovog sučelja sličan je nadzornoj ploči. Svi senzori su grupirani i prikazani s obzirom u kojoj se sobi nalaze. Senzorski podaci su prikazani na vremenskom grafu koji prikazuje kako su se vrijednosti pojedinačnog senzora mijenjali u vremenu. Korisnik ima mogućnost izabrati do kojeg vremena u prošlosti želi prikazivati podatke na grafu. U padajućem izborniku, koji se nalazi kod svakog grafa, može izabrati da se prikazuju podaci dohvaćeni zadnjih sat, šest sati, dvanaest sati ili pak dvadeset i četiri sata od trenutnog vremena.



Slika 3.7 Sučelje za prikaz povijesnih podataka

Prelaskom preko linije grafa mišem, korisniku iskače mali prozor sa prikazom točnih podataka (savršeno točno vrijeme i vrijednost senzora) u toj točki u vremenu (Slika 3.8 Tooltip)





Slika 3.8 Tooltip

### 3.7.1. Izvedba u kodu

Način primitka podataka sa druge komponente odvija se na isti način kao i u komponenti nadzorne ploče:

```
const { state } = useLocation();
```

Objekt `state` se destrukture i sadrži sve podatke vezane za senzore, sobe i podjele po sobama. Za grupiranje senzora po sobama koristi se `groupBy` funkcija iz biblioteke *lodash*:

```
const sensorsByRoom = groupBy(state.array, 'roomName');
```

Rezultat je objekt gdje su ključevi imena soba, a vrijednosti su polja senzora. U svakoj komponenti na web aplikaciji koristi se još jedno stanje komponente koje još nije bilo opisano u prijašnjim poglavljima:

```
const [anchorEl, setAnchorEl] = useState(null);
```

Ovaj dio koda stvara lokalno stanje za komponentu. `anchorEl` je varijabla stanja koja čuva DOM element koji se koristi kao sidro za pozicioniranje izbornika u zaglavlju, a `setAnchorEl` je funkcija koja omogućava ažuriranje tog stanja.

```
const handleMenuOpen = (event) => {
  setAnchorEl(event.currentTarget);
};

const handleMenuClose = () => {
  setAnchorEl(null);
};
```

Kod 3.14 funkcije za kontrolu izbornika

Funkcije `handleMenuOpen` i `handleMenuClose` koriste se za ažuriranje stanja `anchorEl`, što zauzvrat kontrolira otvorenost izbornika. Kada se gumb izbornika klikne, poziva se `handleMenuOpen` s događajem koji sadrži referencu na trenutni gumb, koji se koristi kao sidro za izbornik. Kada se izbornik treba zatvoriti, poziva se `handleMenuClose` koja postavlja `anchorEl` na `null`, zatvarajući izbornik.

Sljedeći dio koda prolazi kroz svaku sobu i njezine senzore:

```
{Object.entries(sensorsByRoom).map(([roomName, sensors]) => (...)
```

Za svaku sobu se prikazuje naziv sobe, a zatim generira mreža koja prikazuje povijesne podatke svakog senzora u toj sobi koristeći samoizrađenu React komponentu `HistoryChart` (Kod 3.15 `HistoryChart` komponenta)

```
<HistoryChart ipAddress={state.data} entityId={sensor.id}
unit={sensor.unit} bearer = {state.bearer} />
```

Kod 3.15 `HistoryChart` komponenta

## 3.8. HistoryChart komponenta

`HistoryChart` je samoizrađena React komponenta koja koristi biblioteku *Recharts* za stvaranje interaktivnih vremenskih grafova u pregledniku. *Recharts* je biblioteka koja omogućuje stvaranje komponenti za prikazivanje podataka, koristeći *D3.js* za pružanje snažnih mogućnosti vizualizacije podataka, dok istodobno održava jednostavnost Reacta.

Dohvaćanje povijesnih podataka iz Home Assistanta odvija se unutar `useEffect` kuke koristeći se REST zahtjevom `GET` (Kod 3.2).

Biblioteka *date-fns* koristi se za rad s datumima i vremenima. Funkcija `sub` koristi se za računanje vremenskog raspona za povlačenje podataka prema korisnikovoj želji (Kod 3.16 Funkcija `sub`).

```
const start = sub(end, {
  [timeRange.endsWith("h") ? "hours" : "days"]: Number(
    timeRange.slice(0, -1)
  ),
});
```

Kod 3.16 Funkcija `sub`

Taj vremenski raspon se pamti u stanju komponente `timeRange`. Komponenta `LineChart` (Kod 3.17) koristi podatke pohranjene u stanju za prikazivanje grafikona data koje je dobio od GET zahtjeva u `useEffect` kuki. Koriste se `CartesianGrid`, `XAxis`, `YAxis` i `Tooltip` komponente za stvaranje mreže, osi i alata za prikaz podataka na grafikonu.

`CartesianGrid` je komponenta koja predstavlja mrežu koordinatnog sustava na kojoj se prikazuje grafikon. Stvara horizontalne i vertikalne linije koje predstavljaju x i y osi, čineći grafikon preglednijim. `strokeDasharray` prop određuje stil crtanih linija mreže, u ovom slučaju pružajući izgled crtane linije.

`XAxis` i `YAxis` su komponente koje predstavljaju x i y osi u grafikonu. Ove komponente pružaju važne informacije o podacima koji se vizualiziraju, poput raspona vrijednosti i vremenskih točaka. Posebno je važno napomenuti da `tickFormatter` prop u `XAxis` komponenti koristi funkciju za formatiranje vremenskih oznaka, pretvarajući ih u lako čitljiv format.

`Tooltip` komponenta je jedan od ključnih elemenata koji poboljšavaju interaktivnost grafikona. Alatni savjeti pružaju dodatne informacije o specifičnoj točki na grafikonu kada korisnik pređe mišem preko nje. U našem kodu, koristi se prilagođena komponenta alatnog savjeta, `CustomTooltip`, koja prikazuje vremensku oznaku i vrijednost odabrane točke. Ova komponenta povećava korisničko iskustvo pružajući relevantne informacije na zahtjev.

```
<LineChart
  data={data}
  margin={{
    top: 5,
    right: 30,
    left: 20,
    bottom: 5,
  }}
>
  <CartesianGrid strokeDasharray="3 3" />
  <XAxis
    dataKey="timestamp"
    angle={-30}
    textAnchor="end"
    tickFormatter={formatXAxis}
    height={70}
```

```

    />
    <YAxis domain=[["dataMin - 10", "dataMax + 10"]] />
    <Tooltip content={<CustomTooltip unit={unit} />}
allowEscapeViewBox={{ x: true, y: true }} />

    <Line type="monotone" dataKey="value" dot={false} />
  </LineChart>

```

Kod 3.17 LineChart komponenta

### 3.9. Opis ispitivanja studijskih slučajeva

Funkcionalnost izrađene web aplikacije testirana je spajanjem Philips Hue pametnog svijetla na React aplikaciju. Zbog toga što je studijski slučaj trebalo provesti u laboratorijskom okruženju FER-a trebalo se ostvariti da za ispitivanje ne bude potrebno previše vremena. Za početak, definirana je automatizacijska skripta unutar konfiguracijske datoteke Home Assistanta. Zadatak skripte bio je uspostava automatske promjene svjetline pametne svjetiljke s obzirom na trenutačno vrijeme. Skripta je automatizirala svjetiljku tako da se svjetlina mijenja četiri puta unutar svakih sat vremena u danu. Tako je ostvarena mogućnost testiranja funkcionalnosti aplikacije unutar samo sat vremena.

Prvo je trebalo pokrenuti Home Assistant instancu na Raspberry Pi uređaju. Nakon toga, trebalo se spojiti na tu instancu preko osobnog računala s IP adresom Raspberry Pi uređaja i dodavanjem točnog porta (8123) na kojem je instanca pokrenuta na uređaju. Važno je napomenuti kako je bilo nužno da su Raspberry Pi i osobno računalo trebali biti spojeni na istu Wi-Fi mrežu.

Nakon uspješnog spajanja, pokrenuta je React aplikacija. Na početnoj konfiguracijskoj stranici unesen je autorizacijski token, IP adresa Home Assistant instance, entitetski identifikator (*entity\_id*) i sve ostale potrebne informacije o svjetiljci (Naziv svjetiljke, mjerna jedinica i naziv sobe). Nakon toga informacije o svjetlini svjetiljke mogle su se bez problema vidjeti na nadzornoj ploči stranice. Prvih petnaest minuta svjetiljka je pokazivala stopostotnu svjetlinu. Automatizacijska skripta je nakon toga promijenila svjetlinu i to je brzo bilo vidljivo u Home Assistant nadzornoj ploči. Za promjenu svjetline na grafikonu u web aplikaciji trebalo je pričekati oko pet sekundi zbog toga što je u kodu definirano da se vrijednosti senzora osvježavaju svakih pet sekundi zbog čišćeg učitavanja podataka.

Pritiskom na prekidač na web stranici, gotovo trenutno, svjetlina Philips Hue svjetiljke pala je na nula posto. Ponovnim klikom na prekidač, svjetiljka je ponovno zasjala.

Ispitivanjem studijskog slučaja utvrdili smo da aplikacija radi točno onako kako je bilo i planirano. Također je utvrđeno da aplikacija ima potencijala za unaprjeđivanje dodavanjem novih funkcionalnosti kao što su kontrola rgb boja svjetiljke, direktna kontrola točnog postotka svjetline i više ponuđenih vrsta grafikona za vizualnu prezentaciju podataka.

Na isti način smo testirali senzore za temperaturu, vlagu i tlak, no ovog puta smo koristili virtualne senzore za testiranje. Virtualni senzori se definiraju u konfiguracijskoj datoteci Home Assistanta. Sastoji se od naziva senzora i definicije ponašanja. Kako je senzor virtualan i simuliran, trebalo je uspostaviti da se ponaša što sličnije pravom senzoru.

Ovakvim ispitivanjem smo samo dodatno utvrdili ono što smo saznali ispitivajući stranicu sa Philips Hue svjetiljkom.

## Zaključak

Iz dana u dan pametni domovi postaju svakodnevnicom svakog od nas. To je robustno tehnološko tržište koje će godinu kroz godinu imati sve veću ulogu. S time se javlja i potreba za što boljom kontrolom nad uređajima u okviru pametnog doma. Ovaj rad je istražio potencijal korištenja biblioteke React u sklopu Javascripta u olakšavanju pristupa brojnim podacima koje pametni dom donosi sa sobom. Zbog svoje jednostavnosti i modularnosti, React ima izniman potencijal, ne samo u razvoju alata za vizualizaciju podataka, već i u područjima razvoja interaktivnih sučelja s kojim se pametni dom može prilagoditi bilo čijim potrebama i preferencijama. Koristeći Home Assistant kao premosnicu između fizičkih IoT uređaja, on ostvaruje sigurnu konekciju i komunikaciju sa IoT uređajima. Kao takav, trebao bi se više koristiti u područjima izrade web aplikacija sa personaliziranim sučeljima u okviru pametnih domova.

# Literatura

- [1] Jen Clark, *What is the IoT? Everything you need to know about the Internet of Things right now*, IBM (2016, listopad). Poveznica: <https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/>
  - [2] Ralph Heredia, *4 Layers of IoT Architecture Explained*, ZipitWireless (listopad, 2022). Poveznica: <https://www.zipitwireless.com/blog/4-layers-of-iot-architecture-explained>
  - [3] Mark K. Pratt, *Top 12 most commonly used IoT protocols and standards*, TechTarget (2022, ožujak). Poveznica: <https://www.techtarget.com/iotagenda/tip/Top-12-most-commonly-used-IoT-protocols-and-standards>
  - [4] Bob Proctor, *Comparing Mesh, Star & Point-To-Point Topology In IoT Networking*, Link Labs (2018, siječanj). Poveznica: <https://www.link-labs.com/blog/iot-topology>
  - [5] Christian Henke, *What Is IoT Security? Risks, Examples, and Solutions*, Emnify (2023, veljača). Poveznica: <https://www.emnify.com/iot-glossary/iot-security>
  - [6] *Internet of things*, Wikipedia. Poveznica: [https://en.wikipedia.org/wiki/Internet\\_of\\_things](https://en.wikipedia.org/wiki/Internet_of_things)
  - [7] *Smart home automation – 7 use-case scenarios in an IoT (Internet of Things) world*, eGlu. Poveznica: <https://www.myeclu.com/smart-home-automation-7-use-case-scenarios-in-an-iot-internet-of-things-world/>
  - [8] Payal Paul, *10 Main Core Concept You Need to Know About React*, Medium (listopad, 2020). Poveznica: <https://payalpaul2436.medium.com/10-main-core-concept-you-need-to-know-about-react-303e986e1763>
- 
- 1) Slika 1.2 ( <https://www.lairdconnect.com/iot-devices/lorawan-iot-devices>)
  - 2) Slika 1.3 ([https://techcrunch.com/2014/10/14/august-smart-lock-onsale/?guccounter=1&guce\\_referrer=aHR0cHM6Ly93d3cuZ29vZ2x1LmNvbS8&guce\\_referrer\\_sig=AQAAAFuTCoGMaQc9UzeYjwgRFRwvxw2OBx5To\\_DkvscV8zQNKNVECKYAT7OCqiTVsQUBUvNI-q3LulPJilih7oBHjKAsMXbxUVfmv6YKcmuIoQW0MbnDEiww3zqceslilyLBC-dHGRBcZ3OHtd6bY\\_SQnwasHF2DgxG6QBx3INc3i7aS](https://techcrunch.com/2014/10/14/august-smart-lock-onsale/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2x1LmNvbS8&guce_referrer_sig=AQAAAFuTCoGMaQc9UzeYjwgRFRwvxw2OBx5To_DkvscV8zQNKNVECKYAT7OCqiTVsQUBUvNI-q3LulPJilih7oBHjKAsMXbxUVfmv6YKcmuIoQW0MbnDEiww3zqceslilyLBC-dHGRBcZ3OHtd6bY_SQnwasHF2DgxG6QBx3INc3i7aS))
  - 3) Slika 1.4 (<https://azbigmedia.com/business/smart-traffic-lights-being-installed-in-greater-phoenix/>)
  - 4) Slika 2.1 (<https://edition.cnn.com/cnn-underscored/reviews/philips-hue-starter-kit>)
  - 5) Slika 2.2 (<https://uxdesign.cc/analyzing-facebooks-major-redesign-from-website-to-app-53ad2f88b0de>)
  - 6) Slika 2.3 (<https://scotthelme.co.uk/setting-up-https-for-home-assistant/>)





## Sažetak

Ovaj završni rad fokusira se na izradu web aplikacije za prikaz podataka u okruženju pametnog doma. Svrha aplikacije je omogućiti korisnicima uvid u senzorske podatke i upravljanje uređajima spojenim na Home Assistant platformu, koja se koristi za automatizaciju doma. Rad je organiziran u tri dijela. U prvom dijelu pružen je uvod u koncept pametnog doma i Internet stvari (IoT). Drugi dio rada fokusira se na Home Assistant platformu i React JavaScript biblioteku koje će se koristiti u izradi aplikacije. Treći dio opisuje implementaciju web aplikacije koristeći React, detaljno opisuje arhitekturu i funkcionalnosti aplikacije te naglašava važnost primjene najnovijih tehnologija u području IoT-a. Kroz stvarni studijski slučaj, predloženo rješenje će se testirati i prikazati njegova praktična primjena u kontekstu pametnog doma. Cilj rada je stvoriti funkcionalnu i korisnički prijateljsku web aplikaciju koja će unaprijediti iskustvo korisnika u pametnom domu.

**Ključne riječi:** Internet stvari, Home Assistant, grafikoni, React, REST API, pametni dom, Philips Hue, senzori, pametna rasvjeta

## Summary

This thesis focuses on the development of a web application for data visualization in the context of a smart home. The purpose of the application is to provide users with insights into sensor data and enable them to control devices connected to the Home Assistant platform, which is used for home automation. The thesis is structured into three parts. The first part provides an introduction to the concept of a smart home and the Internet of Things (IoT). The second part focuses on the Home Assistant platform and the React JavaScript library, which will be used in the implementation of the application. The third part describes the implementation of the web application using React, provides a detailed explanation of the architecture and features of the application, and emphasizes the importance of leveraging cutting-edge technologies in the IoT domain. Through a real-world case study, the proposed solution will be tested and its practical application in the context of a smart home will be demonstrated. The goal of this thesis is to create a functional and user-friendly web application that enhances the user experience in a smart home environment.

**Keywords:** Internet of Things, Home Assistant, charts, React, REST API, smart home, Philips Hue, sensors, smart lighting