Izrada aplikacije za upravljanje pametnom kućom

Svetić, Paulina

Sveučilište u Rijeci

Odjel za informatiku

Preddiplomski sveučilišni studij

Paulina Svetić

Sirius – aplikacija za upravljanje pametnom kućom

Završni rad

Mentor: izv. prof. dr. sc. Sanja Čandrlić

Rijeka, 2021.

Sadržaj

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 18.6.2021.

Zadatak za završni rad

Pristupnik: Paulina Svetić

Naziv završnog rada: Izrada aplikacije za upravljanje pametnom kućom

Naziv završnog rada na eng. jeziku: Development of a smart home management application

Sadržaj zadatka:

Zadatak je studenta osmisliti i razviti aplikaciju koja može poslužiti korisniku za upravljanje pametnom kućom. Student će opisati razvojno okruženje, dokumentirati razvoj te predstaviti izrađenu aplikaciju.

Mentor

Izv. prof dr. sc. Sanja Čandrlić

Voditelj za završne radove

Doc. dr. sc. Miran Pobar

Zadatak preuzet: 18.6.2021.

Paulina Svetić

(potpis pristupnika)

Sažetak

Zadatak ovog završnog rada bio je napraviti programsko rješenje za upravljanje pametnom kućom. Rješenje omogućuje upravljanje više pametnih uređaja korištenjem istog sučelja. Za postizanje tog rezultata korištena je platforma Firebase, tehnologije Vue.js i Quasar, razvojno okruženje IntelliJ IDEA, te alat za analizu koda ESLint.

Rezultat je progresivna web aplikacija, čije će značenje biti kasnije opisano, koja korisniku omogućava rad s uređajima u pametnoj kući: dodavanje novih uređaja, sortiranje uređaja, te upravljanje samim uređajima. Također postoji i statistika koja korisniku omogućuje pregled koliko često je koristio pojedini uređaj.

Motivacija ovog rada leži u tome da se pametni uređaji koriste sve više u svakodnevnom životu, te se ovim putem cilja na automatizaciju određenih funkcija pametnih uređaja kao i *remote* upravljanje uređajima. Grupa korisnika kojoj je namijenjena ova aplikacija su oni korisnici koji imaju mnogo pametnih uređaja oko sebe u svakodnevnom životu te im se na ovaj način olakšava upravljanje tim uređajima.

Ključne riječi: Progresivna web aplikacija, model-view-viewmodel arhitektura, single-page aplikacija, open-source program, *child* element

Uvod

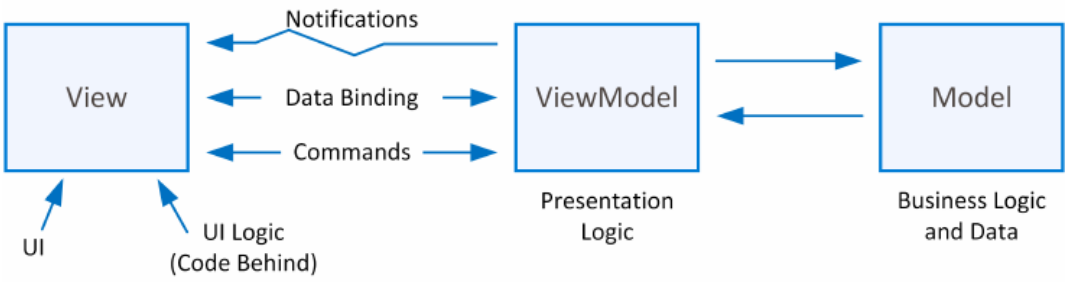
„**Pametna kuća** je sustav koji se sastoji od podatkovne mreže koja spaja i integrira ključne električne uređaje, te dopušta da se oni kontroliraju iz središnjeg izvora. Električni uređaji i funkcije uključuju, no nisu ograničeni, na stvari kao što su grijanje, osvjetljenje, alarmni sustavi, (...), daljinski upravljači i komunikacijski uređaji. Pametne kuće dopuštaju interakciju između kontroliranih elemenata, te upravo ovo sustav čini različitim od običnih kontrolnih sustava okoline. (...) Osnovna instalacija pametne kuće se lako programira da bi udovoljila svim potrebama pojedinca, te je stoga vrlo prilagođena načelima dizajna za svakoga. Sustavi pametne kuće uključuju kontrolu od strane korisnika, automatske funkcije kontrole kroz interakciju između komponenti i veze i kroz interakciju s različitim komunikacijskim medijima.“ – Rade Sanjin, 2015.

Za upravljanje pametnim kućama poželjno je koristiti aplikacije koje omogućuju pristup pojedinačnim uređajima. Za potrebe ovog rada razvijena je progresivna web aplikacija za upravljanje pametnom kućom te opisan proces njenog razvoja.

„Donedavno su postojala dva principa razvoja aplikacija za mobilne uređaje, a to su izvorne, odnosno nativne aplikacije i web aplikacije. Svaka od tih opcija donosi neke prednosti, ali i nedostatke od kojih niti jedna nije zanemariva. U pokušaju eliminacije nedostataka pojavio se novi trend razvoja aplikacija za mobilne uređaje, a to su **progresivne web aplikacije** (PWA). (...) Tvorac ovog novog trenda razvoja aplikacija za mobilne uređaje, odnosno PWA je sam Google (...) – Težak Filip, 2019.

U nastavku su definirani osnovni pojmovi relevantni za ovaj rad.

Model-view-viewmodel je arhitektura koja odvaja pogled (eng. *view*) od modela, odnosno odvaja dio programa koji korisnik vidi i nad kojim vrši interakciju, od logičke strane programa koja obrađuje korisnikove naredbe. Pogled i model ne komuniciraju direktno već preko viewmodel-a (slično kao kontroler u MVC i presenter u MVP arhitekturi). Viewmodel čeka promjenu na strani pogleda te šalje relevantne podatke modelu, model zatim vrši operacije nad dobivenim podacima te ih šalje nazad na viewmodel, koji ih dalje prosljeđuje na pogled gdje se oni prikazuju korisniku na ekran.



Slika 1 - struktura MVVM arhitekture (Patrzyk et al, 2015)

„**Single page aplikacije** su web-aplikacije s jednom stranicom (lokacijom) po kojoj se dinamički navigira i po potrebi mijenja sadržaj. Ovakve aplikacije su popularne zbog kompaktnosti i toga što klijentu pružaju osjećaj desktop aplikacija. Lokacija web-aplikacije se gotovo nikada ne osvježava. Osjećaj prave navigacije i mijenjanja linkova se postiže bibliotekama za preusmjeravanje.“ – Vidović Ana, 2018.

Child element je u HTML strukturi pozicioniran direktno ispod referenciranog elementa. Primjer:

```
<div class=“row“>                                   - parent element
  <div class=“col justify-between“>                 - children elementi (svi q-card elementi)
    <q-card/> <q-card/> <q-card/> <q-card/>
  </div>
</div>
```

Tehnologije korištene u radu

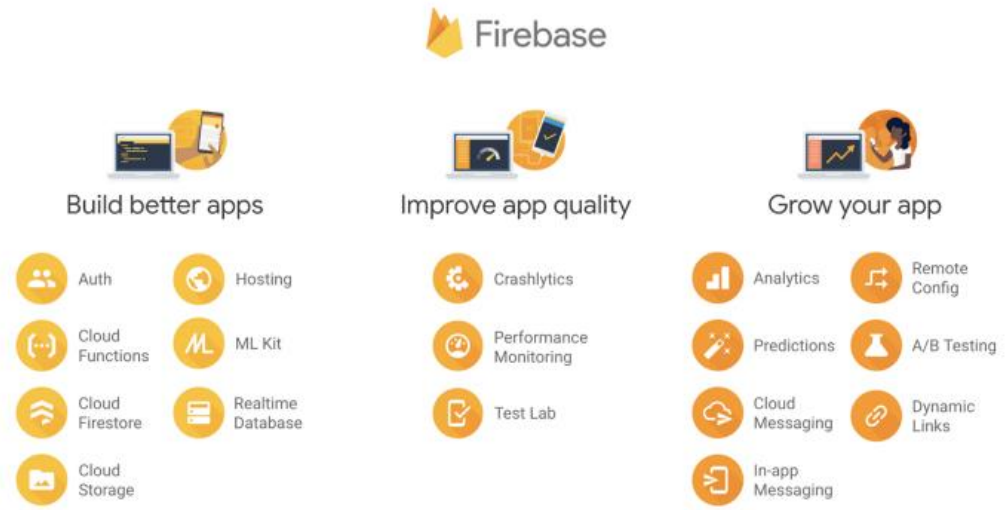
Trendovi i tehnologije za web razvoj se mijenjaju iz dana u dan. Nedostaci starih tehnologija se eliminiraju, a dobri aspekti se nadograđuju. Od svih tih tehnologija izabran je Firebase jer sadrži više različitih usluga koje su bile bitne za ostvarivanje ovog projekta. Izabran je i Vue.js koji je vrlo jednostavan za učenje i rad te ima dobru dokumentaciju i korisničku podršku.

IntelliJ IDEA

IntelliJ IDEA je integrirano razvojno okruženje (eng. IDE – *integrated development environment*) za razvoj računalnih softvera. Razvijen je 2001. godine od strane Jetbrains-a kao jedno od prvih integriranih razvojnih okruženja za Javu. Pruža podršku i za druge jezike, neki od njih su SQL, HTML, Javascript.

Firebase

Firebase[10] je platforma koju je razvio Google, a koristi se za kreiranje, unaprijeđivanje i održavanje mobilnih i web aplikacija. Na Slici 2 su prikazane sve usluge koje nudi Firebase, od kojih su se za izradu ovog projekta koristile usluge hostinga, autentifikacije i baze podataka Firestore.



Slika 2 - usluge koje nudi Firebase (Stevenson, 2018.)

Vue.js

Aplikacijski okviri služe za ubrzanje razvoja web-aplikacija te gotovo svaki programski jezik ima svoj aplikacijski okvir. Najčešće imaju tri sastavna sloja – prezentacijski, aplikacijski (poslovna logika) i podatkovni.

„**Vue.js**[9] je progresivni aplikacijski okvir prvenstveno namijenjen izradi korisničkih sučelja, ali je od početka građen na način da ostavlja dovoljno prostora za nadogradnju i prilagodbu. Jezgrena biblioteka (eng. core library) se fokusira na prezentacijski sloj i lako se kombinira s drugim bibliotekama, pa i drugim projektima. Vue.js je pogodan za izradnju modernih SPA aplikacija (eng. single-page application).“ – Vidović Ana, 2018.

Quasar Framework

Quasar framework je Vue.js aplikacijski okvir otvorenog koda (eng. open-source) za izradu aplikacija koje se postavljaju na web kao single-page ili progresivne web aplikacije.

Korištene **klase** predefinirane u Quasaru[8] :

q-pt-md	*q* je prefiks, *pt* označava gornji padding, *md* označava srednju veličinu paddinga
bg-blue-gray-4	Pozadina koja će biti određene boje, u ovom slučaju plavo-siva
q-gutter-sm	Element sa q-gutter klasom će na svoje children elemente dodati padding gore i lijevo
q-dialog-plugin	Stvara novi prozor koji se otvori kada je ispunjen određeni uvjet
row, col	Kako bi niz elemenata bio smješten u jedan red, potrebno je obuhvatiti ih u zajednički parent element sa klasom ***row***, a svaki individualni element mora imati klasu ***col***
items-center	Svi children elementi elementa sa klasom items-center će po vertikalnoj osi biti pozicionirani na sredini
justify-between	Između svih children elemenata će biti podjednak razmak
text-bold text-h5	Element s ovom klasom će biti podebljani naslov veličine 5

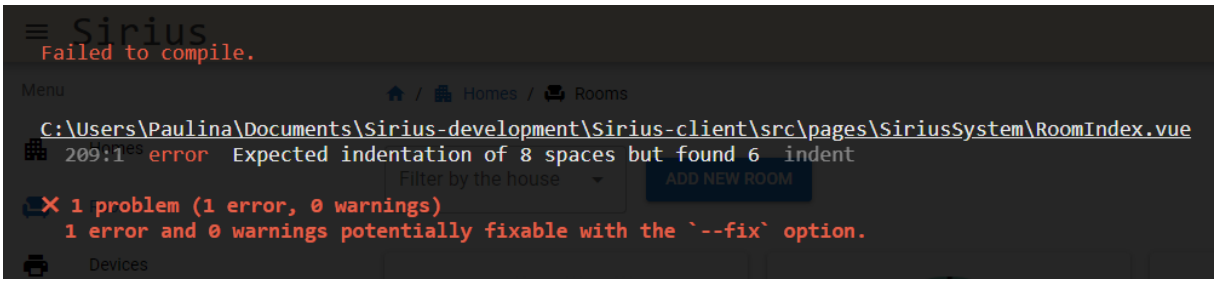
ESLint

ESLint[12] je alat za analizu koda i prepoznavanje dijelova koda koji krše određena pravila. Pravila se mogu mijenjati prema potrebi korisnika.

Neki od primjera tih pravila su:

- indentacija mora biti točno 1 tab (ili 4 razmaka), što pridonosi ujednačenom i preglednom kodu
- stringovi moraju imati jednostruke navodndike (eng. *single-quotes*)

Cijela lista pravila može se naći na službenim stranicama ESLint alata: https://eslint.org/docs/rules/ , 20.09.2021.



Slika 3 - primjer prikaza error poruke uz korištenje ESLint alata

Razvoj aplikacije

Ideja za izradu ove aplikacije razvila se iz razmišljanja o temi *Internet of Things* (skraćeno IoT). IoT je relativno novi pojam, a označava komunikaciju uređaja međusobno povezanih putem internet veze. IoT uređaji mogu biti razni, od dječje igračke sve do dostavnog kamiona.

Nakon generalne ideje trebalo je suziti izbor te je izabran aspekt iz svakodnevnog života – pametna kuća. Na tržištu postoje mnoge aplikacije za upravljanje pametnim uređajima, ali mnoge od njih imaju određena ograničenja, npr. Samsung Smart TV aplikacija, kao što i samo ime govori, je predviđena za upravljanjem isključivo pametnim TV-om tvrtke Samsung. Sirius aplikacijom se nastoji riješiti taj problem na način da se omogući dodavanje svih uređaja bez ograničenja.

Prije samog razvoja aplikacije i kodiranja napravljen je dijagram načina korištenja (eng. *use-case dijagram*) kojim su unaprijed definirane funkcije koje će imati krajnji proizvod. Dijagram je napravljen uz pomoć besplatnog alata StarUML.

Dijagram načina korištenja

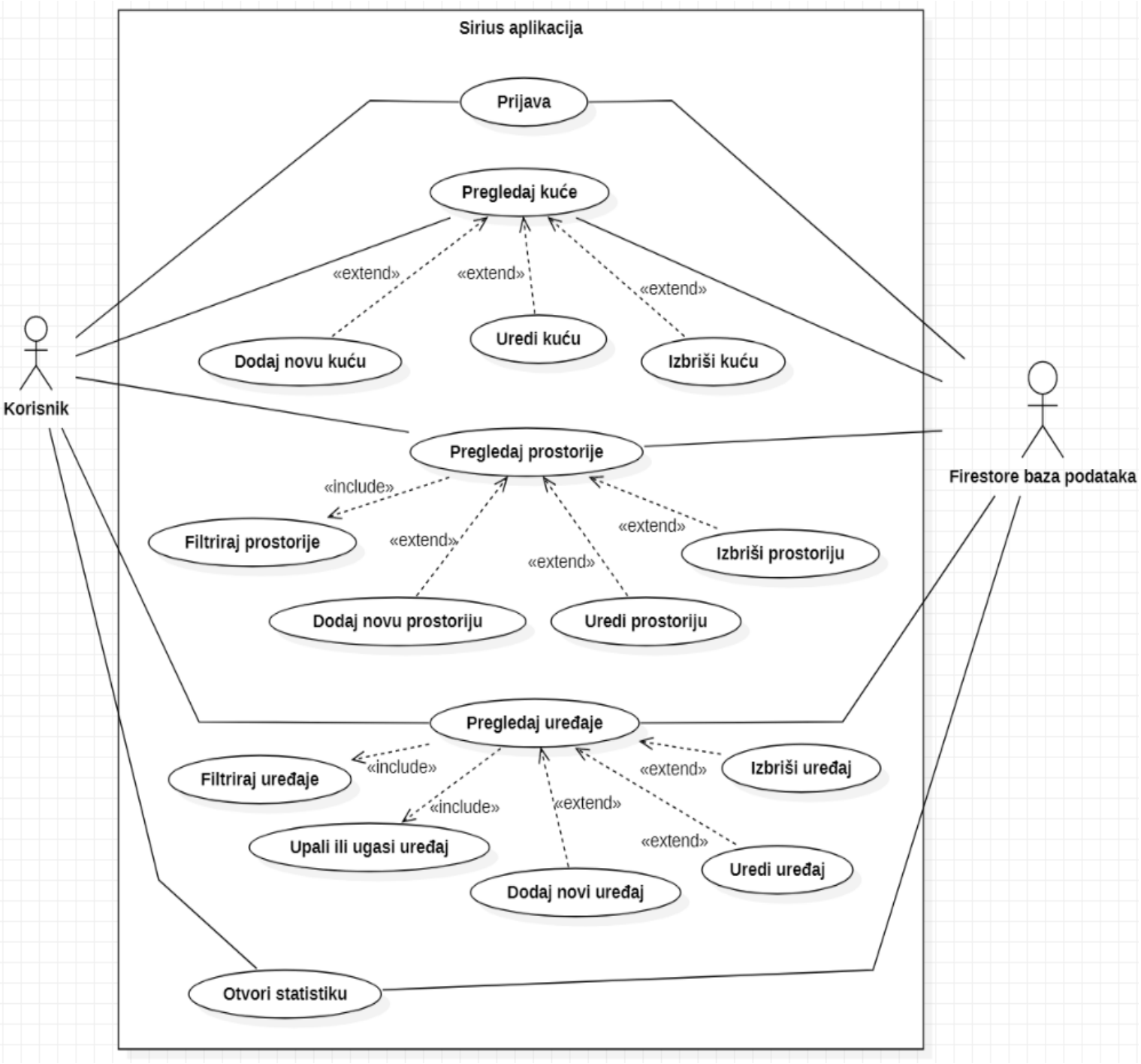
Dijagram načina korištenja prikazan je na slici 4.

Ideja je da se budućom aplikacijom pokriju 3 glavna dijela: virtualne kuće, virtualne prostorije i uređaji. Svaki od tih dijelova aplikacije je odvojen te ima svoju stranicu. Kada je na stranici za virtualne kuće, korisnik osim pregledavanja svih trenutno spremljenih virtualnih kuća u bazi podataka ima opcije dodati novu virtualnu kuću, urediti postojeću virtualnu kuću i izbrisati postojeću virtualnu kuću.

Na stranici za pregled virtualnih prostorija spremljenih u bazi podataka korisnik ima iste opcije kao i s virtualnim kućama – ali ima i dodatnu opciju filtriranja. Filtriranjem korisnik može odabrati po jednu virtualnu kuću iz koje onda dobiva pregled svih virtualnih prostorija.

Na stranici za pregled uređaja koje je korisnik spremio u bazu podataka ponovno postoje sve spomenute opcije, te je dodana glavna opcija aplikacije – upravljanje uređajima.

Osim navedenog, aplikacija ima opcije prijave i odjave, te pregled statistike.



Slika 4 - use case dijagram za Sirius aplikaciju

Proces kodiranja i testiranja

Nakon izrade **dijagrama načina korištenja** stvorila se jasnija ideja o tome kako sama aplikacija treba izgledati. Zatim se projekt podijelio na **manje segmente** te se počelo s programiranjem jednog po jednog segmenta. Segmenti su: naslovna stranica, login, meni, kuće, prostorije, uređaji i statistika.

Idući korak bio je postavljanje **razvojnog okruženja**. Quasar projekt se prvi put generira unutar integriranog razvojnog okruženja naredbom *quasar create Sirius-development*, a svaki idući put se pokreće naredbom *quasar dev*.

Za hosting, bazu podataka i autentifikaciju aplikacije korišten je Firebase. Za spajanje vlastitog Quasar projekta sa Firebase uslugama potrebno je pratiti upute na Firebase stranicama koje su vrlo jasno i detaljno navedene.

Nakon što je napravljeno projektno stablo može se započeti sa **kodiranjem**. Prvo je izrađena naslovna stranica, no ona nema nikakvu funkcionalnost. Na naslovnu stranicu je zatim dodan login gumb, koji na klik poziva Firebase funkciju *SignInWithEmailAndPassword*. Ako su login podaci ispravni korisnika se preusmjerava na stranicu *Homes*, gdje onda ima pristup cijeloj aplikaciji.

Zatim su napravljene stranice Homes, Rooms i Devices, koje su po strukturi kôda vrlo slične, te na kraju statistika.

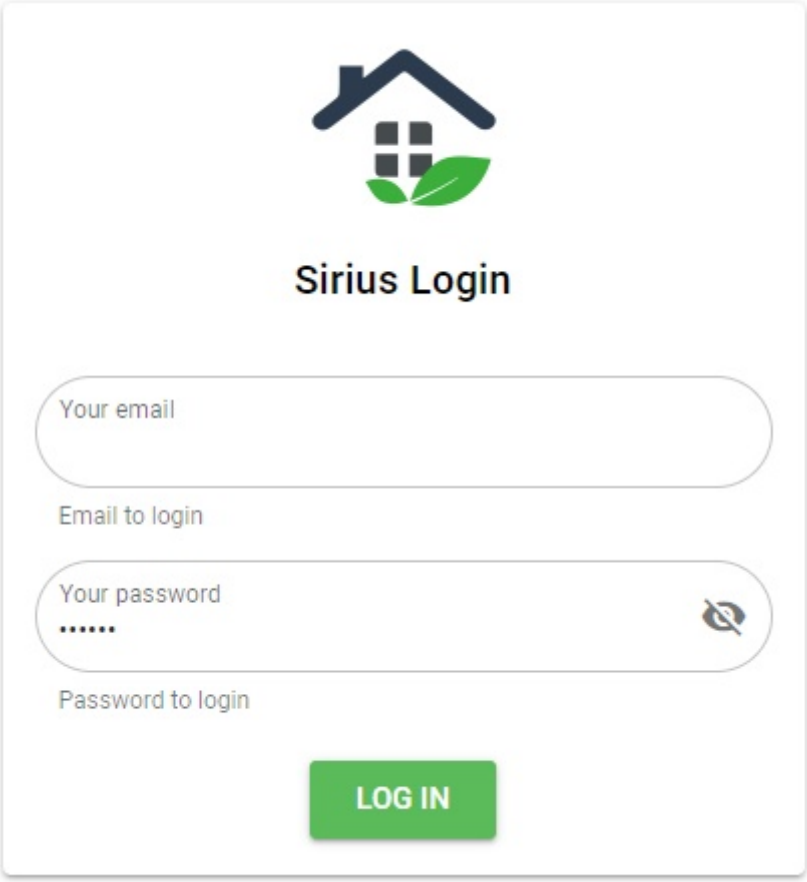
Testiranje se u procesu izrade ove aplikacije odrađivalo paralelno sa samim kodiranjem. To znači da bi se odmah nakon implementacije nove funkcionalnosti provelo testiranje radi li ona kako treba sama za sebe i u odnosu na druge već ranije implementirane funkcije.

Sirius aplikacija

Za rad aplikacije nužan je pristup internetu.

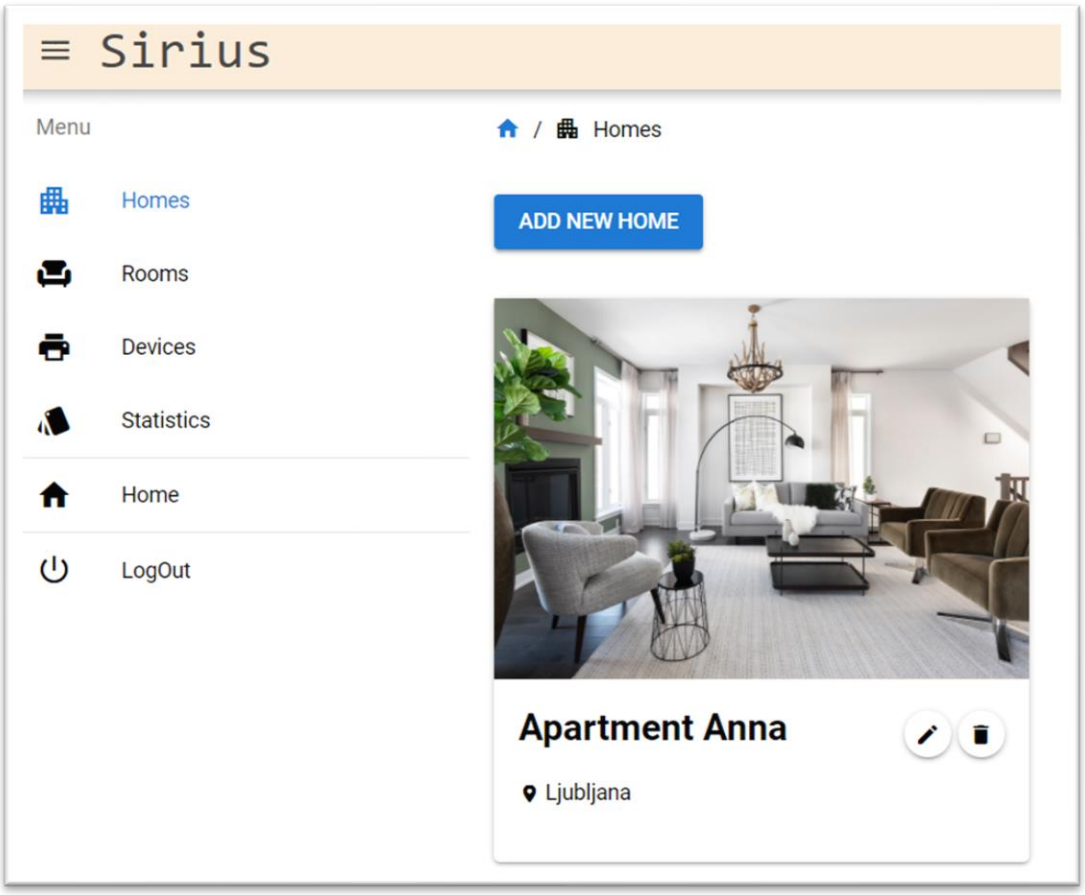
Izbornik se sastoji od linkova za Kuće, Prostorije i Uređaje. Također postoji i stranica s grafom za statistiku, koja prati korisnikove trendove, i Log out gumb.

Za početak rada potrebno je prijaviti se u aplikaciju. Za autentifikaciju korisnika korišten je Firebase, te se kroz njegovo sučelje može manipulirati korisnicima koji imaju pravo prijave u aplikaciju. Sučelje za prijavu unutar Sirirus aplikacije je prikazano na slici 5.



Slika 5 - sučelje za prijavu korisnika u aplikaciju

U fokusu cijele aplikacije su pametni uređaji, ali linkovi izbornika su logički poredani. Prvo je potrebno napraviti virtualnu kuću, što znači kliknuti na *Add new home* gumb i unijeti tražene podatke o kući. Podaci koji se traže su ime kuće, lokacija i slika, a oni služe isključivo za korisnikovo navigiranje po aplikaciji te filtriranje uređaja u aplikaciji. Izgled stranice *Homes* je prikazan na slici ispod.

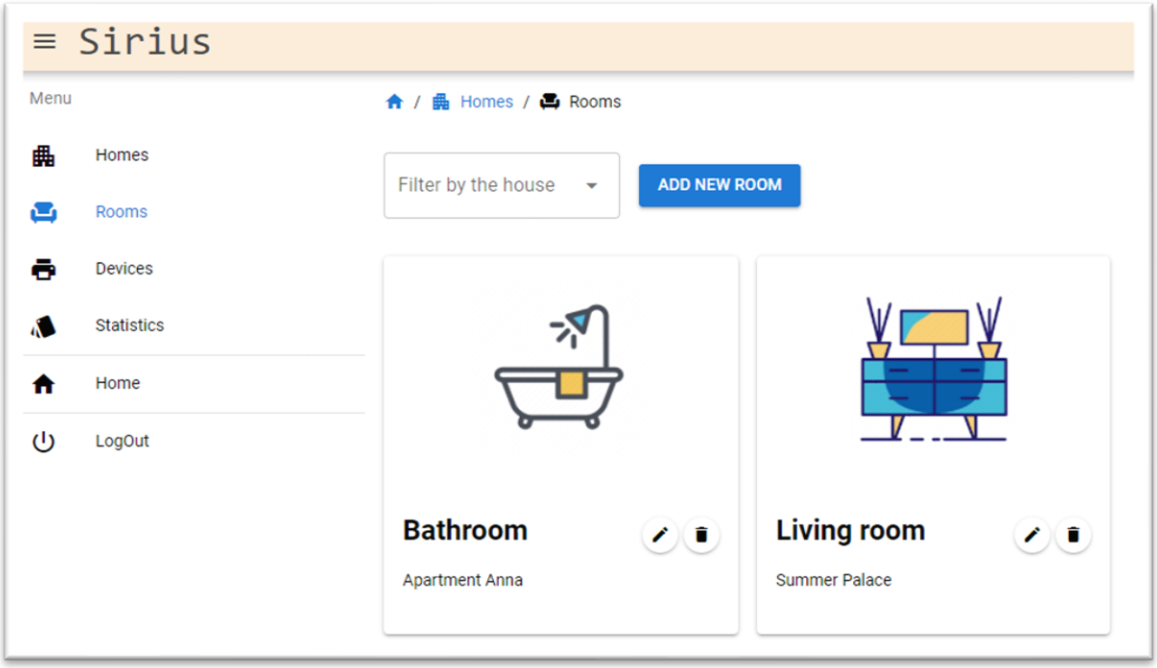


Slika 6 - izgled stranice Homes

Korisnik ima mogućnost napraviti koliko god virtualnih kuća želi.

Nakon što je napravio virtualne kuće, korisnik na idućoj stranici Prostorije radi virtualne prostorije te ih povezuje sa ranije napravljenim virtualnim kućama. To radi klikom na *Add new room* gumb te ponovno unosi tražene podatke. Podaci o prostoriji su ime, ikona ili slika te virtualna kuća kojoj prostorija pripada, a koju bira iz dropdown liste svih ranije napravljenih virtualnih kuća. Također postoji opcija za filtriranje po virtualnim kućama, o čemu će biti više govora kasnije u radu.

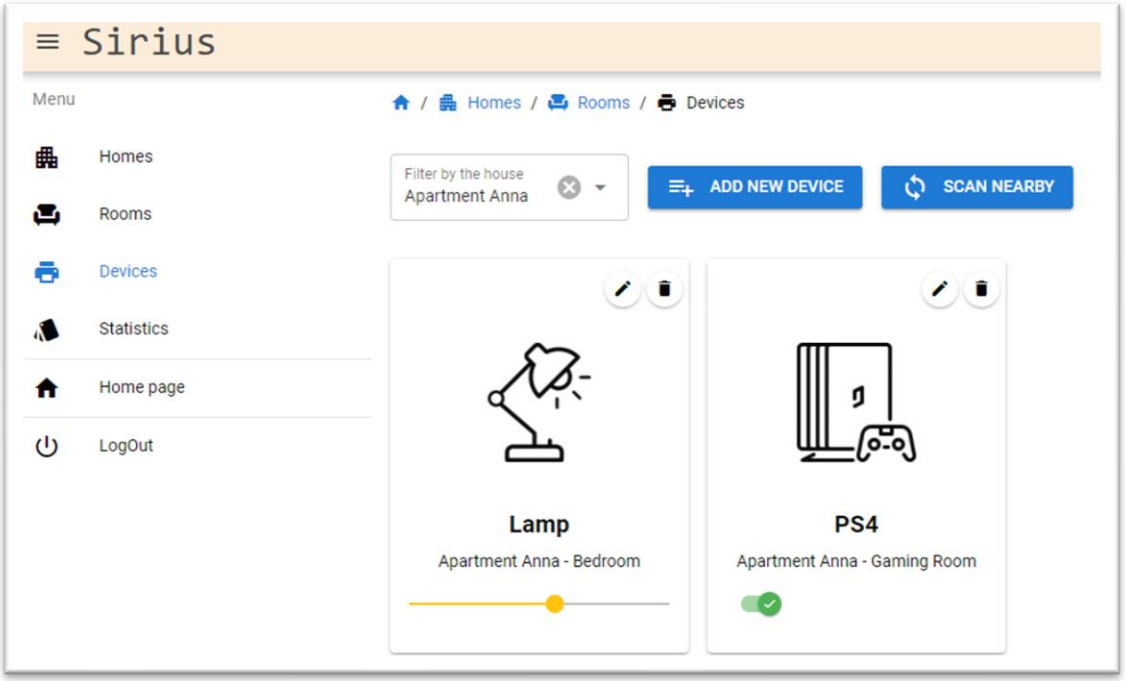
Izgled stranice *Rooms* prikazan je na slici 7.



Slika 7 - izgled stranice Rooms

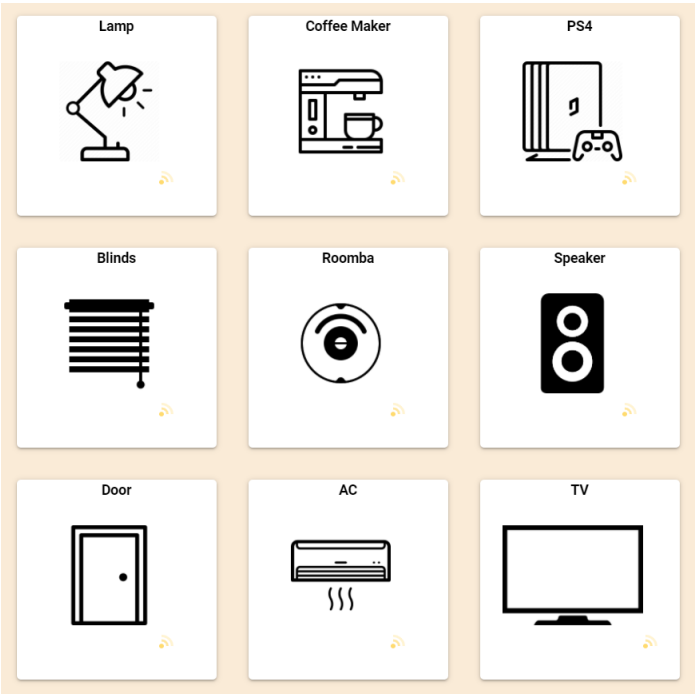
Nakon što je stvorio virtualne prostorije korisnik na stranici Uređaji može dodati svoje uređaje. To ponovno čini klikom na *Add new device* gumb, ali ovaj put ima i opciju skeniranja uređaja u svojoj blizini putem Wi-fi veze*, za što je potrebno kliknuti na *Scan for devices* gumb. Izgled stranice *Devices je prikazan na Slici 8.*

*Pošto nije implementiran dio sa spajanjem na Wi-fi, uređaji se trenutno čitaju iz json datoteke. Ta funkcionalnost će biti detaljnije objašnjena kasnije u radu.



Slika 8 - izgled stranice Devices

Ako je korisnik odabrao opciju skeniranja uređaja otvara mu se sučelje (slika 9) sa svim pronađenim uređajima u blizini te on bira kojeg želi dodati u aplikaciju. Tada se otvara forma sa već ispunjenim podacima (ime uređaja, ikona uređaja i vrsta uređaja) te jedino što korisnik treba odabrati je u koju virtualnu prostoriju želi spremiti taj uređaj. U slučaju da je korisnik kliknuo *Add new device* gumb otvorit će mu se ista forma, ali neće biti ispunjena podacima.

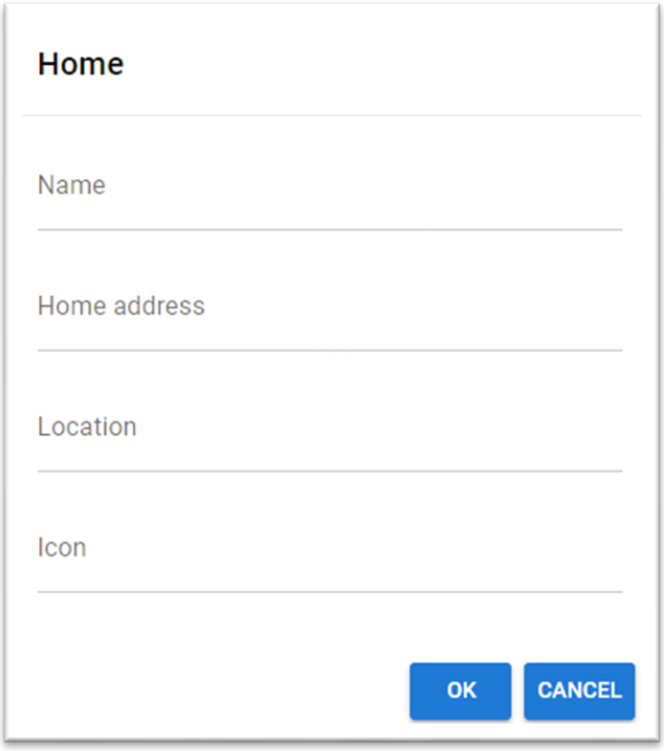


Slika 9 - sučelje koje se otvara klinom na "Scan for devices"

Objašnjenje rada aplikacije

Glavni izgled aplikacije sastoji se od zaglavlja i menija. U zaglavlju je naziv aplikacije „Sirius“ te gumb sa otvaranje i zatvaranje menija. Meni se sastoji od ranije navedenih linkova: Kuće, Prostorije, Uređaji te Statistika.

Pri otvaranju stranice *Homes* iz baze podataka se čitaju sve napravljene virtualne kuće te se one prikazuju korisniku kao što je prikazano na slici 6. Korisnik može promijeniti podatke o svakoj od napravljenih virtualnih kuća klikom na gumb za edit (ikona olovke). Tada mu se otvara isto sučelje kao kada je stvarao virtualnu kuću (slika 10), ali sa ispunjenim podacima za odabranu kuću koje onda može promijeniti. Korisnik može i obrisati napravljenu virtualnu kuću klikom na gumb za brisanje. Virtualna kuća se tada briše iz baze te nije moguće ponovno dohvatiti te podatke.



Slika 10 - sučelje za dodavanje nove virtualne kuće

Prelazak na stranicu *Rooms* moguć je pritiskom na odgovarajući link u meniju ili klikom na jednu od virtualnih kuća. Ukoliko korisnik odabre opciju iz menija otvorit će mu se popis svih soba iz svih kuća, primjer sa dvije sobe je prikazan na slici 7. Nakon toga korisnik može filtrirati prikazane sobe po virtualnoj kući u koju su smještene. Da je korisnik u kartici *Homes* kliknuo na jednu od virtualnih kuća, otvorila bi mu se kartica *Rooms* sa već primijenjenim filterom za odgovarajuću virtualnu kuću.

Pri otvaranju stranice *Rooms* čitaju se podaci o napravljenim virtualnim sobama iz baze podataka, te se bez primijenjenog filtera prikazuju sve sobe. Filtriranje je implementirano pomoću select polja i url-a. Url služi za komunikaciju između kartice *Homes* i *Rooms*. Ako je url formata:

http://localhost:8080/#/rooms?selectedHouse=Apartment%20Anna

znači da će se u select polju primijeniti filter za kuću imena Apartment Anna. Korisnik može odabrati drugi filter u select polju ili očistiti filter da bi mu se prikazale sve sobe iz svih virtualnih kuća, ali tada se url neće promijeniti.

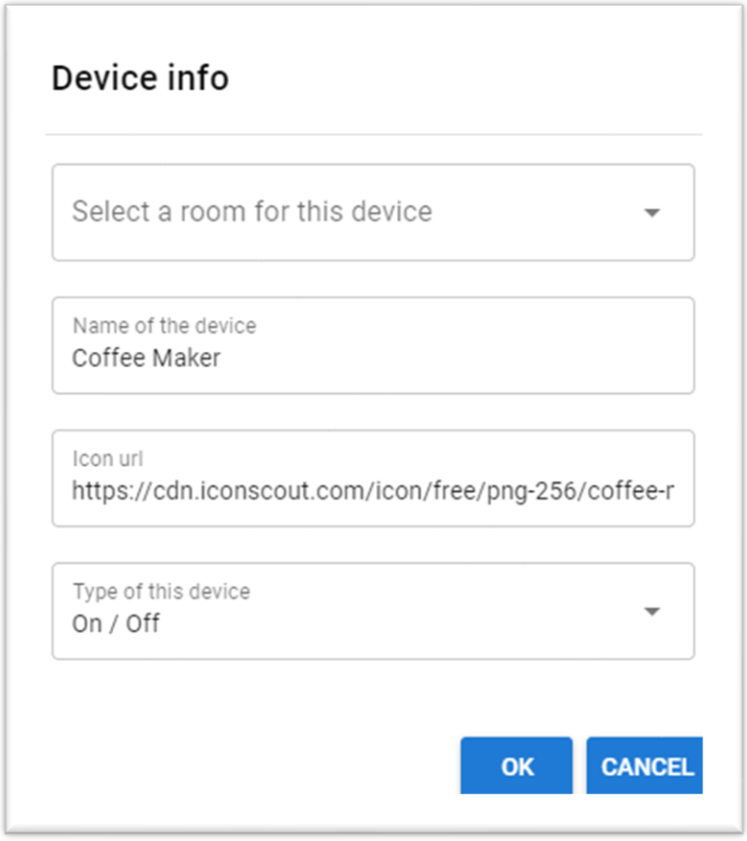
Korisnik i ovdje ima mogućnost promjene podataka za već napravljene virtualne sobe, te za brisanje virtualnih soba iz baze podataka. Kada korisnik izbriše određenu virtualnu kuću sve virtualne sobe povezane na tu kuću neće biti obrisane jer bi se na taj način obrisali i svi pripadajući uređaji. Pod pretpostavkom da korisnik još uvijek posjeduje sve uređaje te da će ih nastaviti koristiti pomoću Sirius aplikacije, na ovaj način ih može samo „premjestiti“ u neku drugu virtualnu sobu.

Prelazak na stranicu *Devices* također je moguć na dva načina, putem linka u meniju ili klikom na određenu sobu na stranici *Rooms*. Popis uređaja je prikazan kao na Slici 8. Filter na kartici *Devices* funkcionira na isti način kao i na kartici *Rooms*.

Osim toga, prozor *Devices* ima dodatnu opciju *Scan nearby* koja prikazuje sve uređaje u blizini spojene na istu Wi-fi mrežu na koju je spojen korisnik. Trenutno je ta opcija implementirana na način da se iz json datoteke dohvaćaju predefinirani uređaji, a struktura jednog od tih uređaja prikazana je na slici 11. Uređaji su iz json datoteke na ekran prikazani kao na slici 9, te se pritiskom na jedan od uređaja otvara sučelje za dodavanje novih uređaja, ali sa ispunjenim podacima kao što je prikazano na slici 12. Korisnik bira sobu za odabrani uređaj, može promijeniti postojeće podatke ako to želi, te pritiskom na gumb OK stvara odabrani uređaj.


```
[
  {
    "UIDDevice": null,
    "id": "1",
    "name": "Lamp",
    "icon": "https://cdn0.iconfinder.com/data/icons/office-and-business-29/85/19_kight_black-512.png",
    "type": "Slider",
    "sliderValue": 50,
    "currentState": false,
    "lastChange": "",
    "numberOfChanges": 0
  },
```


Slika 11 - struktura uređaja "Lampa" iz json datoteke



Slika 12 - prozor za dodavanje novog uređaja putem Scan nearby opcije

Zadnje polje pri dodavanju novog uređaja je *Tip uređaja* i postoje dvije mogućnosti za odabir, prva je On/off – što znači da uređaj u pitanju ima mogućnost samo paljenja i gašenja, npr. Play Station, aparat za kavu; i druga opcija *Slider* što znači da uređaj ima funkciju rada u većem rasponu, npr. svijetlo koje može svijetliti bilo kojom jačinom, a ne samo biti upaljeno ili ugašeno, ili zvučnici koji mogu biti pojačani do neke razine.

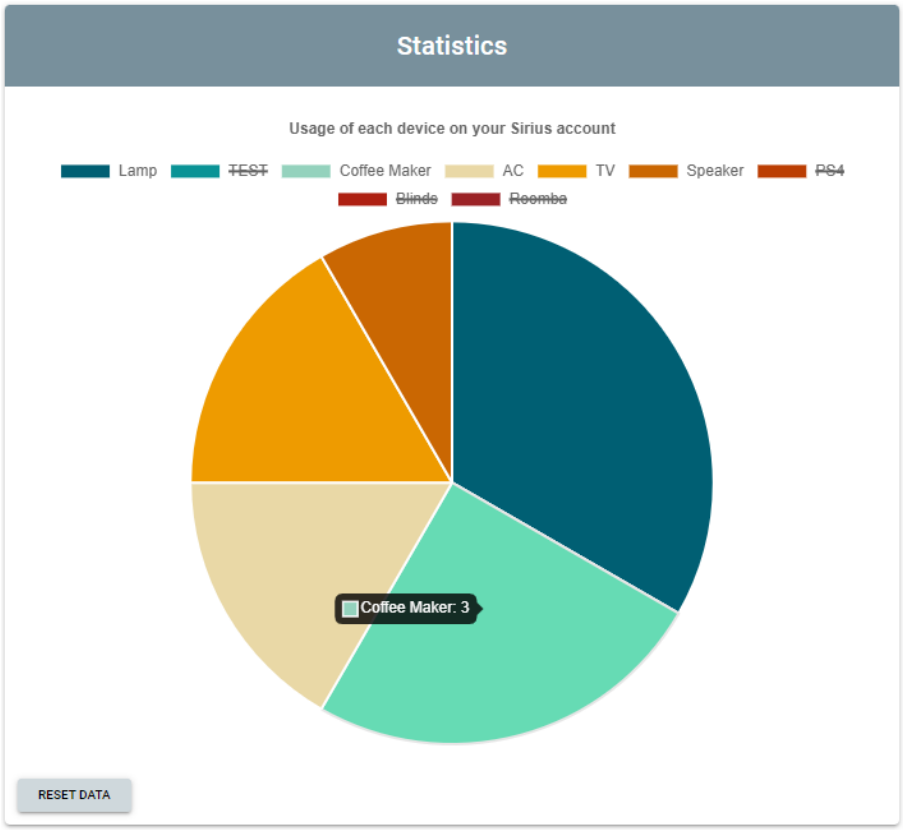
Statistika i kôd za statistiku

Za izradu i prikaz statistike u aplikaciji korištena je biblioteka (eng. *library*) Chart.js[11]. Ovaj dio aplikacije je posebno izdvojen jer osim čitanja podataka iz baze podataka i prikazivanja korisniku na ekran koristi i dodatne funkcije za računanje učestalosti korištenja svakog pojedinog uređaja.

Statistika je u aplikaciji prikazana Pie chart grafom te prikazuje koliko često je korisnik koristio određeni uređaj. Korisnik može klikom na ime uređaja u legendi maknuti taj uređaj iz statistike (kao što je naveden primjer za Test, PS4, Blinds i Roomba uređaje), te prelaskom miša preko određene boje vidjeti o kojem se uređaju radi (primjer je dan za uređaj Coffee Maker).

Korisniku je omogućeno i filtriranje statistike po virtualnim kućama, a select polje za to se nalazi u aplikaciji odmah iznad pie chart grafa.

Gumb *Reset data* u donjem lijevom kutu služi za poništavanje svih trenutnih vrijednosti statistike iz baze podataka.



Slika 13 - prikaz statistike u aplikaciji

```
deviceTemplate: {
  UIDDevice: null,
  id: this.makeID( length: 5),
  roomID: null,
  name: null,
  type: null,
  icon: null,
  currentState: false,
  sliderValue: 0,
  lastChange: this.getDate(),
  numberOfChanges: 0
},
```

Slika 14 - svi redovi tablice Uređaj

Na slici 14 su prikazana sva polja koja će se spremiti u bazu za svaki novi uređaj:

- *UIDDevice* je UID koji je automatski generirao Firestore.
- *id* je nasumično generiran niz znakova duljine 5.
- *lastChange* je datum u trenutku stvaranja ili bilo kakve promjene uređaja.
- *currentState*, *sliderValue* i *numberOfChanges* će imati vrijednosti kao što je prikazano na slici, a označavaju redom: ako je tip uređaja *On/off*, tada će *currentValue* biti njegovo trenutno stanje, true za On i false za Off; ako je tip uređaja *Slider*, tada će se njegova vrijednost mjeriti na skali od 0 do 100 te će se u polje *sliderValue* spremati njegova trenutna vrijednost; *numberOfChanges* je polje koje se koristi za statistiku – svaki put kada korisnik promijeni vrijednosti *currentState* ili *sliderValue* polja, *numberOfChanges* će se uvećati za jedan.
- *roomID* je *ID* sobe u koju je uređaj postavljen unutar aplikacije
- *name* je ime uređaja koje korisnik upiše za određeni uređaj.
- *type* je tip uređaja koji može biti On/off ili Slider.
- *icon* je link na ikonu uređaja koja će se prikazivati u aplikaciji za određeni uređaj.

```
31  <div class="center-content q-pa-md row items-start q-gutter-md">
32    <!-- for every device in database, render a new q-card -->
33    <q-card v-for="device in filteredDevices" :key="device.id"
34    inline style="width: 250px; height: 330px">
35      <div align="right" style="padding: 5px">
36        <q-btn @click="onUpdateRow(device)" round size="sm" icon="edit" style="margin: 5px" />
37        <q-btn @click="onDeleteRow(device)" round size="sm" icon="delete" />
38      </div>
39      <img alt="Icon of a device." :src="device.icon">
40      <q-card-section>
41        <p class="text-bold text-h6" align="center">{{ device.name }}</p>
42        <p align="center">{{ device.roomID.homeID.name }} - {{ device.roomID.name }}</p>
43        <!-- depending on type of each device (on/off or slider), render q-toggle or q-slider -->
44        <q-toggle
45          @input="updateToggleValue(device, device.id, device.currentState)"
46          v-if="device.type === 'On / Off'"
47          v-model="device.currentState"
48          checked-icon="check"
49          unchecked-icon="clear"
50          color="green"
51        />
52        <q-slider
53          @change="updateSliderValue(device, device.id, device.sliderValue)"
54          v-if="device.type === 'Slider'"
55          v-model="device.sliderValue"
56          :min="0" :max="100"
57          color="amber" />
58      </q-card-section>
59    </q-card>
60  </div>
```

Slika 15 - front end dio aplikacije

Na slici 15 je dio kôda kojim se prikazuje svaki pojedini uređaj u aplikaciji. Na liniji 33 piše *„za svaki uređaj u filteredDevices listi generiraj novi q-card element duljine 250 pixela i visine 330 pixela sa svim ispod navedenim elementima“*. Tako će svaki od generiranih uređaja imati iduće elemente:

Linijama 36 i 37 se prikazuju gumbi za uređivanje i brisanje uređaja.

Linijom 39 se pomoću :src=“device.icon“ prikazuje ikona svakog pojedinog uređaja.

Linijama 41 i 42 se prikazuju ime trenutnog uređaja i pripadna virtualna prostorija i kuća.

Svaki uređaj može biti tipa „On/off“ ili tipa „Slider“. Funkcija *v-if* na linijama 46 i 54 provjerava upravo vrijednost tipa uređaja i na temelju toga generira ili q-toggle element za „On/off“ tip uređaja, ili q-slider element za „Slider“ tip uređaja.

Konačni izgled q-card elemenata za uređaje u aplikaciji kao i razlika između q-toggle i q-slider elemenata se može vidjeti na Slici 8.

```
280   updateSliderValue: async function (device, id, val) {
281     let uid = ''
282     let currentChanges = 0
283     await this.$db.collection( collectionPath: 'devices')
284       .get()
285       .then((rows) => {
286         rows.forEach( callback: (row) => {
287           if (row.data().id === id) {
288             uid = row.id
289             currentChanges = row.data().numberOfChanges + 1
290           }
291         })
292       })
293     await this.$db.collection( collectionPath: 'devices')
294       .doc(uid)
295       .update( data: {
296         sliderValue: val,
297         numberOfChanges: currentChanges
298       })
299   },
```

Slika 16 - funkcija za ažuriranje Slidera

Na slici 16 je prikazana funkcija koja se poziva svaki put kada korisnik promijeni vrijednost na slideru za neki uređaj (prikazano na slici 15, linija 53). Nakon što se dohvate svi uređaji iz baze, pomoću polja *id* se nađe uređaj na kojem je korisnik promijenio vrijednost, te se polje *numberOfChanges* uveća za jedan. Pomoću .update funkcije (linija 295) se u bazi podataka promijeni vrijednost slidera i broj promjena na tom uređaju.

Funkcija za ažuriranje On/off uređaja napravljena je na isti način, ali se u bazi podataka mijenjaju različita polja (*currentState* polje umjesto *sliderValue*).

Zaključak

U okviru ovog završnog rada razvijena je funkcionalna progresivna aplikacija za korištenje na desktopu ili na mobilnom uređaju. Aplikacija je kreirana pomoću Vue.js, Quasar Framework i Firebase alata.

Prije samog razvoja izrađen je dijagram načina korištenja koji je unaprijed definirao funkcionalnosti aplikacije, pa je sa jasno istaknutim ciljevima bilo lakše krenuti s programiranjem.

Krajnji rezultat je aplikacija koja omogućava prijavu korisnika, unos virtualnih kuća i prostorija, te konačno i unos samih uređaja kojima se može upravljati unutar aplikacije.

Za daljni razvoj aplikacije prvi korak bi bio implementirati skeniranje uređaja povezanih na istu Wi-fi vezu. Zatim bi bilo potrebno izmjeniti postojeću bazu podataka, odnosno nadograditi ju tako da podržava autorizaciju više korisnika, pošto se trenutno bazira samo na jednom. Također bi bilo dobro unaprijediti korisničko iskustvo sa manjim promjenama unutar aplikacije, jedan od primjera je uvesti funkciju alarma koju bi korisnik podesio prema svojoj potrebi, npr. ako je vanjska temperatura ispod 15°C upaliti će se grijanje.

Popis izvora

[1] Težak, Filip, „Progresivne web aplikacije“, završni rad, Fakultet organizacije i informatike, Sveučilište u Zagrebu, Varaždin, 2019, dostupno na: https://repozitorij.foi.unizg.hr/islandora/object/foi%3A5612/datastream/PDF/view , zadnji put posjećeno: 20.09.2021.

[2] Vidović, Ana, „Klijentske web aplikacije i Vue.js“, diplomski rad, Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu, 2018, dostupno na: https://repozitorij.pmf.unizg.hr/islandora/object/pmf%3A6070/datastream/PDF/view , zadnji put posjećeno: 20.09.2021.

[3] Rade, Sanjin, „Pametna kuća“, diplomski rad, Filozofski fakultet u Rijeci, Odsjek za politehniku, Sveučilište u Rijeci, 2015., dostupno na: https://repository.ffri.uniri.hr/islandora/object/ffri%3A817/datastream/PDF/view , zadnji put posjećeno: 23.09.2021.

[4] Patrzyk Joanna, Patrzyk Bartlomiej, Katarzyna Rycerz, Marian Bubak, „Towards a novel environment for simulation of quantum computing“, Computer Sciene (1) 2015, dostupno na: https://www.researchgate.net/figure/The-Model-View-ViewModel-MVVM-architectural-pattern-In-MVVM-the-View-layer-is_fig3_275258051 , zadnji put posjećeno 23.09.2021.

[5] Stevenson, Doug, „What is Firebase? The complete story, abridged.“, dostupno na: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0 , zadnji put posjećeno: 23.09.2021.

[6] Ranger, Steve, „What is the IoT? Everything you need to know about the Internet of Things right now“, dostupno na: https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/ , zadnji put posjećeno: 20.09.2021.

[7] Hearn, Patrick, „4 Best Smart Home Apps for Smart Home Automation“, dostupno na: https://www.online-tech-tips.com/smart-home/4-best-smart-home-apps-for-smart-home-automation/ , zadnji put posjećeno: 20.09.2021.

[8] https://quasar.dev/ , zadnji put posjećeno: 20.09.2021.

[9] https://vuejs.org/ , zadnji put posjećeno: 20.09.2021.

[10] https://firebase.google.com/ , zadnji put posjećeno 23.09.2021.

[11] https://www.chartjs.org/ , zadnji put posjećeno: 20.09.2021.

[12] https://eslint.org/docs/rules/ , zadnji put posjećeno: 20.09.2021.