# Prikupljanje i vizualizacija podataka iz sustava pametnih soba

**Premelč, Dominik**

SVEUČILIŠTE U RIJECI

**TEHNIČKI FAKULTET**

Preddiplomski sveučilišni studij računarstva

Završni rad

# Prikupljanje i vizualizacija podataka iz sustava pametnih soba

Rijeka, rujan 2022.

Dominik Premelč

0069085335

SVEUČILIŠTE U RIJECI

**TEHNIČKI FAKULTET**

Preddiplomski sveučilišni studij računarstva

Završni rad

# Prikupljanje i vizualizacija podataka iz sustava pametnih soba

Mentor: Doc. dr. sc. Sandi Ljubić

Rijeka, rujan 2022.

Dominik Premelč

0069085335

## Izjava o samostalnoj izradi rada

Izjavljujem da sam ja, Dominik Premelč, student Tehničkog fakulteta u Rijeci na studiju računarstva , autor završnog rada pod naslovom Prikupljanje i vizualizacija podataka iz sustava pametnih soba. Izjavljujem da sam samostalno izradio ovaj rad pod mentorstvom Doc. dr. sc. Sandija Ljubića.

Rijeka, rujan 2022.

_____

Dominik Premelč

## Zahvala

Zahvaljujem se mentoru Doc. dr. sc. Sandiju Ljubiću na prijedlozima , konstruktivnim savjetima i strpljenju tijekom izrade ovog rada.

# Sadržaj

# Popis slika

# Popis tablica

# Popis ispisa

# 1. Uvod

U današnje moderno doba, tehnologija se primarno koristi s namjerom da čovjek jednostavnije i brže može obavljati svakodnevne aktivnosti. Uporaba „pametnih“ uređaja olakšava nam komunikaciju, snalaženje u prostoru, obavljanje svakodnevnih radnji kao što su kupovina, razonoda, učenje i slično.

Trend porasta uporabe „pametnih“ uređaja odrazio se i u hotelskoj industriji, unutar koje su kombinacijom takvih i sličnih uređaja nastale tzv. „pametne sobe“, koje pružaju dodatnu razinu luksuza i udobnosti nad običnim sobama. Uz dodatnu udobnost, s pametnim sobama dolazi i veliko olakšanje servisnih funkcija. Točnije, kroz nadgledanje stanja pametnih soba, osoblje može lakše primijetiti potrebu za određenim servisima na uređajima, postoji li potreba za čišćenjem sobe ili je pogodnije čišćenje sobe odgoditi za neko drugo vrijeme, ovisno o prisutnosti gostiju u sobi ili mogućim „ne ometaj“ funkcijama. Uz to, vrlo je bitno spomenuti i štednju energije koja se postiže kroz različite optimizacije klimatizacije, grijanja te osvjetljenja u sobi ovisno o prisustvu gosta u sobi. Navedene su samo neke od brojnih prednosti pametnih soba za koje je, kako bi se optimalno mogle iskoristiti, potrebno nadgledati stanja svih uređaja u pojedinim sobama.

U sklopu ovog završnog rada implementirana je aplikacija pomoću koje je moguće vršiti nadziranje podataka iz pametnih soba. Sustav koji je razvijen sastoji se od tri dijela – simulatora pametnih soba, baze podataka i aplikacije za nadzor. Simulator pametnih soba koristi podatke očitane u pametnim sobama jednog zagrebačkog hotela u razdoblju od 2013. godine do 2021. godine, kao i podatke o atmosferskom stanju s Državnog hidrometeorološkog zavoda [6] (u daljnjem tekstu DHMZ), prikupljenih u razdoblju od 2013. godine do 2021. godine. Navedeni podaci zapisuju se u bazu podataka u određenim intervalima te simuliraju očitanja pametnih soba u stvarnom vremenu. Također je moguće zapisati pripremljene povijesne podatke u svrhu korištenja i arhiviranja. Aplikacija za nadzor uzima podatke zapisane u bazi podataka te ih prikazuje kroz sučelje ovisno o kriterijima koje zadaje korisnik. Moguće je pregledati trenutna stanja pojedinih soba, stanja u zadanom rasponu vremena, zatim stanja na određeni dan i slično. Također, moguće je analizirati stanja u sobama na različitim grafovima. Dodatno, stanja soba potkrijepljena su informacijama o atmosferskom stanju iz Državnog hidrometeorološkog saveza. Spona između ove dvije aplikacije je baza podataka koja je dizajnirana tako da zapisivanje u bazu te čitanje iz nje bude što brže i responzivnije.

U ovom radu pobliže će se opisati proces implementacije te način rada navedenog sustava. Prikazat će se mogućnosti sustava, posebice funkcije za nadzor, kako se koriste, na koji način su implementirane te objasniti odluke vezane uz implementaciju i rad baze podataka i simulatora.

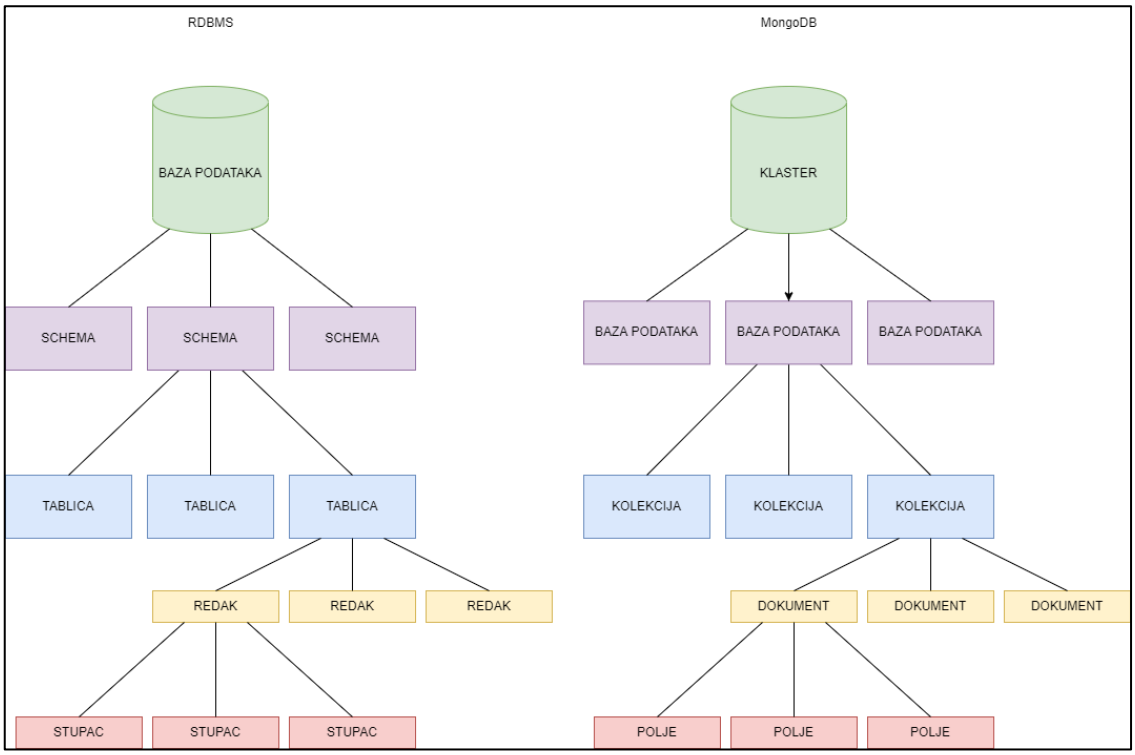
# 2. Korištene tehnologije

Za razvoj sustava korištene su određene aktualne tehnologije. Za poslužiteljsku stranu sustava za nadzor korišten je *SpringBoot. Z*bog svoje jednostavnosti, brzine i mogućnosti implementacije *REST* arhitekture *SpringBoot* se pokazao kao najrobusniji i najbrži okvir za razvoj web aplikacija. Za pohranu podataka korištena je *MongoDB* baza podataka koja je idealna zbog svoje sposobnosti za jednostavno rukovanje velikim količinama podataka. Na korisničkoj strani korištene su *Javascript* knjižnice *Axios* za slanje i primanje *HTTP* zahtjeva te *Highchart* za vizualizaciju podataka pomoću grafova. U nastavku su ukratko opisane navedene tehnologije te na koji način su iskorištene u postupku implementacije.

## *2.1. MongoDB*

*MongoDB* je ne-SQL (engl. *no-SQL*) baza podataka, odnosno nerelacijska baza podataka, koja omogućuje dohvaćanje i pohranu podataka koji nisu usko povezani kao podaci u tablicama relacijskih baza podataka [1].

Relacijske baze svoje podatke spremaju u tablice povezane primarnim i stranim ključevima, dok s druge strane *MongoDB* podatke sprema u dokumente u JSON formatu koji međusobno ne moraju biti povezani. JSON format omogućuje da se pozivamo na vrijednosti u očitanjima na principu ključ-vrijednost te se tako preslikava u objekte u kodu, što uvelike pojednostavljuje prijenos podataka između simulatora, baze i nadzorne aplikacije [1]. Očitanja iz pametnih soba nisu međusobno povezana te se zato ovaj format spremanja i dohvaćanja podataka pokazao kao puno bolji odabir u usporedbi s formatom relacijskih baza. Također, ovakav format je puno pogodniji za rukovanje velikim količinama podataka. Na Slici 2.1. vidimo usporedbu pojmova koji se koriste u relacijskim bazama i MongoDB ne-SQL bazama.




*Slika 2.1. Usporedba pojmova koji se koriste u relacijskim i MongoDB NoSQL bazama*

Dodatno, kao još jedna prednost nerelacijske baze nad relacijskom je fleksibilnost podataka. Podaci zapisani u kolekcijama mogu se mijenjati kroz vrijeme te nisu ograničeni inicijalno postavljenim stupcima kao što je to slučaj kod podataka u tradicionalnim, relacijskim bazama.

Primjerice, može biti slučaj da u određenim hotelskim sobama postoji uređaj koji ne postoji u drugim sobama ili se nakon inicijalnog postavljanja baze podataka u sobe dodao novi uređaj, čija je očitanja također potrebno pratiti. Prilikom korištenja nerelacijskih baza, jednostavno i uz minimalno dorađivanja baze moguće je krenuti s unosom novih dokumenata koji uključuju vrijednosti očitane s novih uređaja.

## *2.2. SpringBoot*

*SpringBoot* je *Java* okvir (engl. *Framework*) otvorenog koda temeljen na mikroservisnoj arhitekturi, a koji je nastao od već postojećeg *Spring* okvira. Omogućuje jako brzo razvijanje *REST* (engl. *Representational State Transfer*) aplikacija zahvaljujući ugrađenim *Tomcat, Jetty* ili *Undertow* aplikacijskim poslužiteljima, automatskoj konfiguraciji i podrškom za *Maven* i *Gradle* alate za upravljanje ovisnostima [2].

*REST* je arhitekturalni stil koji pruža standarde između računalnih sustava na web-u kako bi mogli lakše komunicirati. Također, omogućuje i razdvajanje poslužiteljske od klijentske strane. Komunikacija između klijentske strane s poslužiteljskom svodi se na izmjenu poruka tako da obje strane znaju kakvu poruku moraju primiti te kakvu poruku moraju poslati. Navedene poruke su HTTP (engl. *Hypertext Transfer Protocol*) zahtjevi, čiji su glavni tipovi navedeni u nastavku:

- *GET* – dohvati jedan ili više resursa
- *POST* – stvori novi resurs
- *PUT* – ažuriraj specificirani resurs
- *DELETE* – ukloni specificirani resurs.

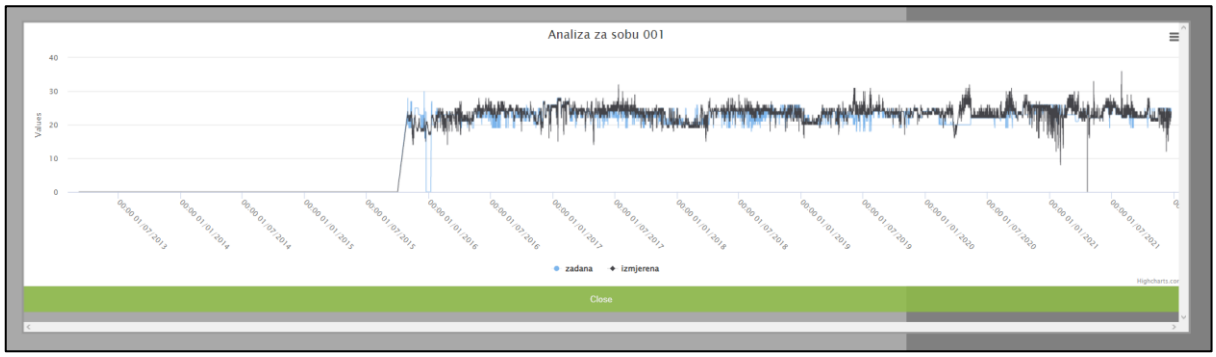
Aplikacija za nadgledanje radi na *REST* principu te se sve operacije za dohvaćanje podataka s poslužiteljske strane rade pomoću HTTP zahtjeva [3]. Poslužiteljska strana na primljene zahtjeve odgovara s odgovarajućim podacima u JSON formatu s kojima klijentska strana radi neovisno o poslužiteljskoj strani.

## 2.3. Ostalo

U razvoju aplikacije za nadgledanje korišteno je nekoliko različitih *Javascript* knjižnica. *Axios* je *Javascript HTTP* klijent baziran na obećanjima (engl. *promise*) koji omogućuje slanje i formatiranje *HTTP* zahtjeva iz pretraživača kao i formatiranje podataka iz *HTTP* odgovora. Aplikacija je bazirana na *REST* principu te A*xios* služi kao alat za dohvat podataka s poslužiteljske strane [4].

*Highcharts* je Javascript knjižnica za implementaciju interaktivnih grafova. Napisana je isključivo u *HTML5* i *Javascript*-u te ne zahtijeva posebne dodatke (engl. *plugins*) za prikaz podataka. Sadrži interaktivne funkcije kao što su zumiranje na grafu i analiza pojedinih točaka na grafu [5].

Na Slici 2.2. vidljiv je linijski graf koji prikazuje vrijednosti zadane i izmjerene temperature u vremenu od 1.1.2013. godine do 31.12.2021. godine (sveukupno 1,829,254 točaka). Zbog svog *boost* modula, *Highchart* je idealan za vizualizaciju podataka u aplikaciji za nadgledanje jer čak i u slučaju ovako velikog skupa podataka, graf ostaje responzivan te se svaka od skoro dva milijuna točaka može pregledati.




*Slika 2.2 Prikaz zadane i izmjerene temperature u rasponu 2013. do 2021. godine*

*Highchart* također omogućuje dinamično dodavanje točaka na graf što ujedno predstavlja ključan zahtjev za prikaz podataka u stvarnom vremenu. Kod aktivnog nadziranja pametne sobe potrebno je grafički prikazati stanja svih uređaja u sobi kao i svih očitanih atmosferskih stanja te navedene grafove ažurirati u realnom vremenu. *Highchart* ovaj posao obavlja vrlo učinkovito te svaki od nacrtanih grafova zadržava responzivnost te interaktivne funkcije.

# 3. Dostupni podatkovni skupovi

U svrhu demonstracije funkcija sustava korištena su dva skupa podataka, podaci iz Državnog hidrometeorološkog zavoda i podaci iz stvarnih pametnih soba jednog zagrebačkog hotela. Arhivski podaci unutar aplikacije podudaraju se s oba skupa podataka. Podaci primljeni iz aplikacije za simulaciju pametnih soba su podaci iz prošlosti gdje je vrijeme promijenjeno u vrijeme u trenutku simulacije stanja u pametnoj sobi. Ova promjena omogućuje nadzor svih mogućih stanja unutar soba u stvarnom vremenu, bez potrebe da se prikupljaju aktualni podaci iz fizičke pametne sobe unutar nekog hotela ili sličnog objekta.

## 3.1. DHMZ podaci

Prva skupina podataka dobavljena je od Državnog hidrometeorološkog zavoda. Podaci su meteorološke prirode, izmjereni na području grada Zagreba u vremenskom periodu od 2013. do 2021. godine. U tablici 3.1. navedeni su različiti podaci iz ovog skupa te njihov opis i mjerne jedinice.

*Tablica 3.1 Podaci iz Državnog Hidrometeorološkog zavoda*

| Podatak | Opis | Mjerna jedinica |
|---|---|---|
| Vrijeme | Datum i vrijeme kada su napravljena očitanja | Vremenska značka |
| Zračenje | Količina sunčeve energije koja dospijeva na horizontalnu površinu zemlje | $J/cm^2$ |
| Temperatura | Izmjerena temperatura u vrijeme očitanja izražena u stupnjevima celzijusevim | °C |
| Smjer vjetra | Izmjereni smjer vjetra | Vrijednosti 0 - 32 |
| Brzina vjetra | Izmjerena brzina vjetra | m/s |
| Vlažnost | Izmjerena vlažnost zraka u trenutku očitanja | $Kg/m^3$ |

### 3.2. Podaci iz sustava pametnih soba

Drugu skupina podataka dao je na korištenje i analizu jedan od zagrebačkih hotela. Taj skup sastoji se od podataka očitanih iz pametnih soba hotela u vremenskom periodu od ožujka 2013. godine do prosinca 2021. godine. U Tablici 3.2. navedeni su različiti tipovi podataka iz ovog skupa te njihov opis.

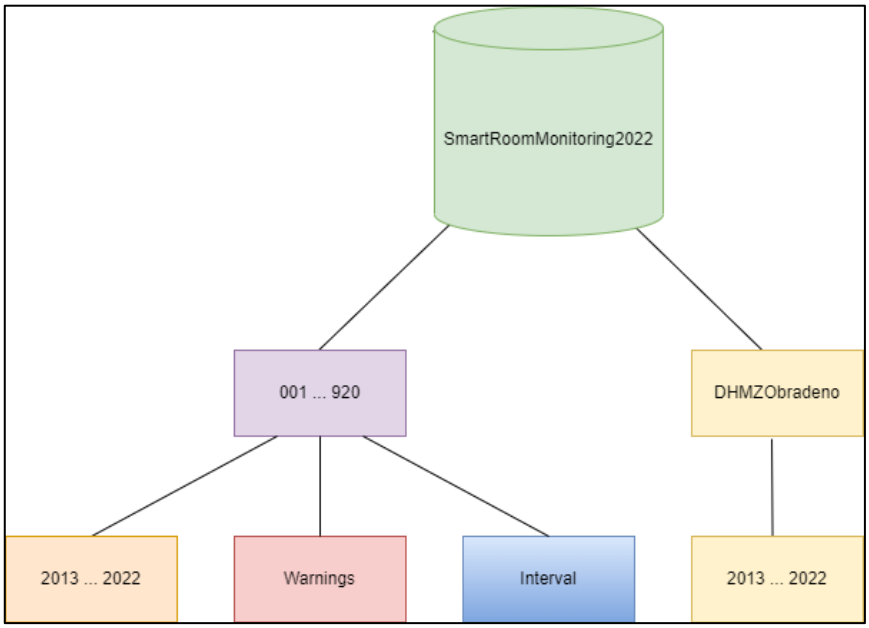
*Tablica 3.2. Podaci iz hotelskih pametnih soba*

| Podatak | Opis | Tip podataka |
|---|---|---|
| Broj sobe | Broj sobe iz koje su iščitane vrijednosti | Broj |
| Vrijeme | Datum i vrijeme kada su napravljena očitanja | Vremenska značka (engl. *timestamp*) |
| Prisutnost | Vrijednost koja označuje je li gost prisutan u sobi | Binarni (0/1) |
| Zadana temperatura | Temperatura zadana na klima uređaju | °C |
| Izmjerena temperatura | Temperatura izmjerena u sobi | °C |
| Status klime | Vrijednost koja označuje je li klima upaljena | Binarni (0/1) |
| Mod klime | Oznaka rada klime | Slovčane oznake za grijanje H i hlađenje C |
| Prozor | Vrijednost koja označuje je li prozor otvoren | Binarni (0/1) |
| Ventil | Otvoren ili zatvoren ventil u HVAC sustavu | Binarni (0/1) |

# 4. Baza podataka

Za bazu podataka korištena je prethodno spomenuta i opisana ne-SQL baza *MongoDB*. Korištena baza sadržana je u cijelosti na oblaku pomoću servisa *MongoDB Atlas*. *MongoDB* grozd (engl. *cluster*) strukturiran je tako da se maksimizira brzina čitanja i zapisivanja podataka u stvarnom vremenu, bez da se značajno uspori brzina čitanja veće količine podataka iz arhiva za analitički dio aplikacije.

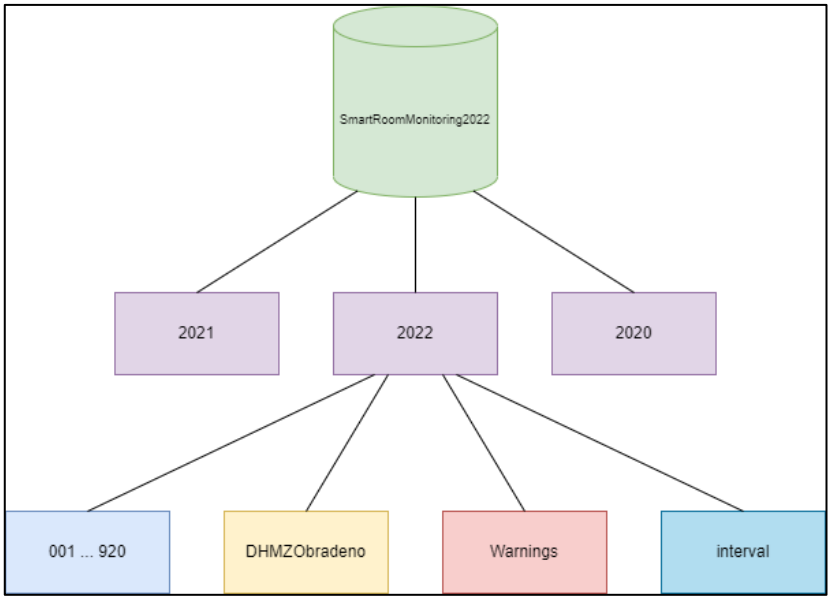
Kroz razvoj aplikacije testirani su razni modeli baze podataka koji su imali svoje prednosti i nedostatke. Kao prvi model korištena je baza u kojoj su sva očitanja pametnih soba u jednoj kolekciji te u drugoj kolekciji sva DHMZ očitanja. Ovaj model pružao je najveću brzinu upisivanja podataka u bazu, međutim, brzo je odbačen jer su i najjednostavniji upiti u bazu zahtijevali mnogo vremena. Sljedeći testirani model podijelio je podatke mjerenja iz pametnih soba prema sobama u kojima su očitani. Navedeni model je značajno ubrzao čitanja iz baze, ali ta ista očitanja i dalje nisu bila dovoljno brza za prikazivanje mjerenja u stvarnom vremenu. Kao jedan od najboljih testiranih modela te najsličniji konačnom modelu istaknuo se model prikazan na slici 4.1.




*Slika 4.1. Vizualni prikaz jednog od testiranih modela baze podataka*

Na najvišoj razini podaci su podijeljeni na baze podataka prema broju sobe te jedna baza podataka za DHMZ podatke. Na razini niže su kolekcije s podacima podijeljenima prema godinama. Ovaj model je radio prihvatljivo za analitičke funkcije kod većih i manjih vremenskih raspona kod

dohvata podataka. Međutim, za unos podataka u stvarnom vremenu dolazi do čestog mijenjanja baze podataka jer podatke unosimo za svaku sobu kod svakog očitanja pa je stoga ovaj model bio neprikladan. Vrijeme za unos podataka iz sobe koja se aktivno nadzire bilo je zadovoljavajuće, ali vrijeme unosa svih ostalih soba u stvarnom vremenu bilo je poprilično veće te bi dolazilo do velikih vremenskih razmaka između očitanja iz prioritetne sobe i očitanja iz svih ostalih soba. Kako bi se dodatno optimizirala čitanja i pisanja podataka u stvarnom vremenu kao i čitanja za analitičke funkcije, testiran je drugi model koji je u konačnici odabran kao odgovarajući model za implementirani sustav. Konačni model je prikazan na slici 4.2.




*Slika 4.2. Vizualni prikaz strukture baze podataka*

Na najvišoj razini, podaci su podijeljeni prema godinama. Tako se izbjegava dugo pretraživanje unutar baze kod čitanja u vremenskim rasponima unutar godinu dana. Također, kod unosa podataka u stvarnom vremenu potrebno je mijenjati samo kolekciju u koju se unosi odgovarajuće očitanje. Unutar svake baze podataka podaci su podijeljeni na kolekcije čiji su nazivi brojevi soba u kojima su napravljena očitanja.

Za odabir konačnog modela baze podataka napravljeni su odgovarajući testovi. U daljnjem tekstu, prvi model označava onaj odbačeni, a drugi model označava onaj konačni. Prvi test koji je napravljen jest test brzine zapisa podataka u stvarnom vremenu. Mjerena je brzina zapisa jednog dokumenta s podacima iz pametne sobe, to jest, sobe koja se aktivno nadzire, brzina jednog zapisa dokumenta s DHMZ atmosferskim podacima te brzina zapisa svih soba osim one koja se aktivno nadzire. Vremena su mjerena istovremeno kako bi dobili podatke za očekivano korištenje

aplikacije. Za ovaj test te sve ostale korišteni su podaci iz devedeset i sedam pametnih soba te svi DHMZ podaci. Rezultati su dobiveni kao prosječno vrijeme zapisivanja i čitanja iz baze podataka nakon tisuću akcija zapisivanja i stotinu akcija čitanja iz baze podataka. Vrijeme u tablici iskazano je u milisekundama. Kraj naslova stupaca zapisana je veličina podataka koji se zapisuju u bazu podataka. Rezultati su vidljivi u tablici 4.1.

*Tablica 4.1. Rezultati testiranja zapisivanja u bazu*

| **Zapis podataka** | Pametna soba – jedan dokument (251b) [ms] | DHMZ – jedan dokument (206b) [ms] | Pametne sobe – dokumenti svih ostalih soba (24.5kb) [ms] |
|---|---|---|---|
| Prvi model | 60.18 | 56.01 | 5641.16 |
| Drugi model | 45.92 | 47.37 | 5183.85 |

Kao što je vidljivo iz rezultata drugi model je učinkovitiji od prvog, ali jedina značajna razlika primjetljiva je kod unosa podataka svih soba, pola sekunde. Međutim, puno veća razlika može se primjetiti kod čitanja podataka. Testirala se brzina dohvata jednog, posljednje unesenog, dokumenta iz pametnih soba i jednog, također posljednje unesenog, iz DHMZ atmosferskih podataka, svi dokumenti u vremenskom rasponu od godine dana za jednu pametnu sobu i atmosferske podatke, isti tip podataka u vremenskom rasponu od 3 godine te naposljetku svi podaci iz jedne pametne sobe i svi podaci o atmosferskim stanjima u vremenskom rasponu od 2013. do 2022. godine. Rezultati ovog testiranja vidljivi su u tablicama 4.2., 4.3., 4.4. i 4.5. Zbog lakše usporedbe, vremena u tablici 4.2. izražena su u milisekundama dok su u tablicama 4.3. , 4.4 i 4.5. izražena u sekundama.

*Tablica 4.2. Rezultati testiranja dohvaćanja pojedinačnih dokumenata*

| **Čitanja podataka** | Pametna soba – zadnji dokument (251b) [ms] | DHMZ – zadnji dokument (206b) [ms] |
|---|---|---|
| Prvi model | 81.23 | 61.41 |
| Drugi model | 62.6 | 56.7 |

*Tablica 4.3. Rezultati testiranja dohvata podataka od jedne godine*

| **Čitanja podataka** | Pametna soba – dokumenti u rasponu od godinu dana (32.62Mb) [s] | DHMZ – dokumenti u rasponu od godinu dana (2.38Mb) [s] |
|---|---|---|
| Prvi model | 30.93 | 2.48 |
| Drugi model | 10.46 | 1.02 |

*Tablica 4.4. Rezultati testiranja dohvata podataka od tri godine*

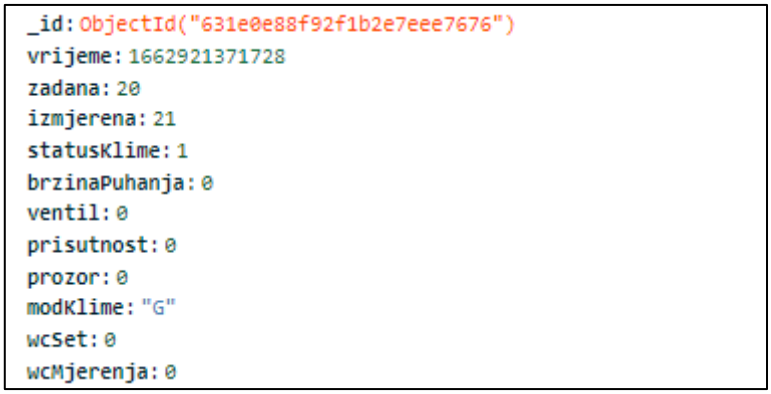
| **Čitanja podataka** | Pametna soba – dokumenti u rasponu od tri godine (98.62Mb) [s] | DHMZ – dokumenti u rasponu od tri godine (7.15Mb) [s] |
|---|---|---|
| Prvi model | 40.93 | 6.53 |
| Drugi model | 32.68 | 3.56 |

*Tablica 4.5. Rezultati testiranja dohvata podataka od 2013. do 2022. godine*

| **Čitanja podataka** | Pametna soba – dokumenti u rasponu od 2013. do 2022. godine (289.94Mb) [s] | DHMZ – dokumenti u rasponu od 2013. do 2022. godine (16.67Mb) [s] |
|---|---|---|
| Prvi model | 115.85 | 15.90 |
| Drugi model | 87.28 | 6.42 |

Iz svih prikazanih rezultata vidljivo je zašto je drugi model izabran za konačnu implementaciju sustava. Zbog načina na koji rade pretrage podataka unutar aplikacije, puno je pogodnije da podatke na višoj razini podijelimo prema godinama umjesto prema broju sobe u kojoj su očitani.

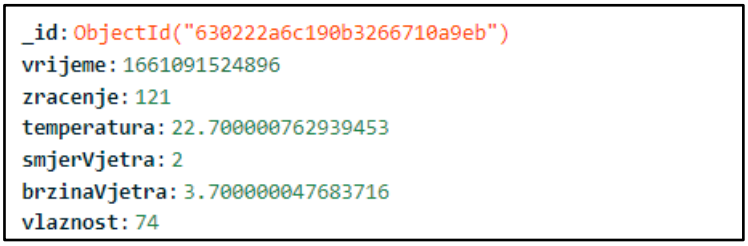
Podaci u MongoDB bazi spremljeni su u obliku različitih dokumenta. Primjeri tih dokumenata prikazani su u nastavku. Na Slici 4.3. prikazan je primjer dokumenta sa stanjem iz pametne sobe.




*Slika 4.3. Primjer dokumenta s podacima očitanima iz pametne sobe*

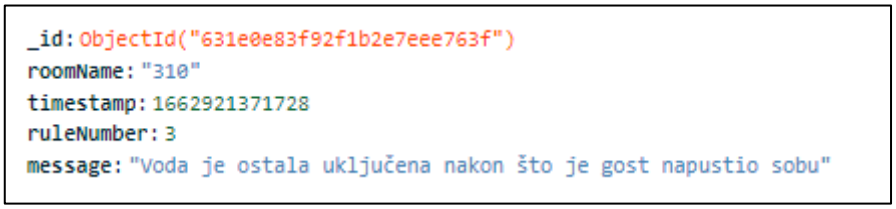
Kao što je vidljivo na Slici 4.3. u dokument se ne sprema broj sobe iz koje je očitanje. Broj sobe ne mora se zasebno zapisivati u svaki dokument zbog načina na koji je strukturiran *MongoDB* grozd. Zbog mogućnosti uklanjanja broja sobe iz dokumenta,uz vrijeme, dodatno štedimo i na prostoru koji zauzima grozd. Unutar svake od baza podataka također se nalaze kolekcije

*DHMZObradeno*, *Warnings* i *Interval*. *DHMZObradeno* sadrži sve podatke očitane unutar odabrane godine. Na Slici 4.4. prikazan je primjer dokumenta s očitanim atmosferskim stanjem.



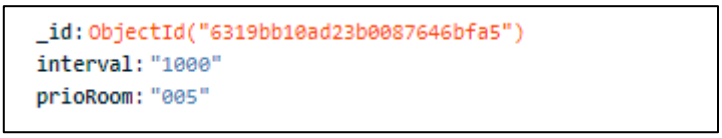

*Slika 4.4. Primjer dokumenta s očitanim atmosferskim podacima*

*Warnings* sadrži sva nerazriješena upozorenja koja su zapisana prilikom otkrivanja nepravilnosti ili nedozvoljenog stanja prilikom čitanja stanja iz pametnih soba. Nedozvoljena stanja, tj. pravila za detekciju istih, definirana su unaprijed te se provjeravaju prilikom svakog unosa podataka u bazu. Na taj su način dostupna korisniku u svakom trenutku, neovisno o tome je li dana soba u određenom trenutku aktivno nadzirana ili nije. Primjer dokumenta s podacima o upozorenju vidljiv je na Slici 4.5.




*Slika 4.5. Primjer dokumenta s podacima o upozorenju*

*Interval* u svakom trenutku sadrži smo jedan dokument u kojem su zapisani podaci potrebni za određivanje intervala čitanja stanja iz pametnih soba kao i broj sobe koja se u tom trenutku aktivno nadzire. Dokument zapisan u kolekciji „interval“ vidljiv je na slici 4.6.




*Slika 4.6. Dokument u kolekciji "interval"*

Ovaj dokument služi kao poveznica između aplikacije za nadzor i aplikacije za simulaciju pametnih soba. Više o načinu kako se koristi bit će opisano u daljnjim poglavljima.

Opisani model baze podataka omogućuje brzo čitanje i pisanje podataka u stvarnom vremenu, dok produljuje vrijeme čitanja podataka samo u velikim vremenskim rasponima. Ovakav kompromis je prihvatljiv jer korisnik već unaprijed očekuje da će čitanje velikog broja podataka u rasponu od više od godinu dana trajati duže.

Podaci iz pametnih soba kao i podaci iz *DHMZObradeno* indeksirani su po parametru vremena koje je unutar baze spremljeno u UNIX formatu. Prilikom prikaza vremena kod korisnika, vrijeme se prilagođava prema lokalnoj vremenskoj zoni. Indeksiranje je potrebno samo po parametru vremena, budući da se svi upiti koje aplikacija radi zasnivaju isključivo na vremenu i broju sobe, pri čemu je broj sobe ujedno i naziv kolekcije kojoj je potrebno pristupiti.

# 5. Aplikacija za simulaciju pametnih soba

Zbog potrebe za demonstracijom funkcija aplikacije za nadzor pametnih soba razvijena je aplikacija za simulaciju pametnih soba. Glavni smisao aplikacije je da oponaša pametne sobe unutar nekog hotela ili sličnog objekta. Na isti način kao što bi se očitavala stanja različitih uređaja u nekoj pametnoj sobi, simulator čita stanja zapisana u CSV datotekama, zatim ih preuređuje u oblik koji je pogodan za zapis u bazu podataka, provjerava te iste podatke s obzirom na nedozvoljena stanja te ih zapisuje u bazu podataka. Pri pokretanju aplikacije, čita se ranije spomenuti dokument zapisan u kolekciji *interval* unutar baze podataka. U sklopu tog dokumenta nalazi se podatak o intervalu u kojem trebamo zapisati očitanja u bazu podataka te broj sobe koja se trenutno aktivno nadzire, tj. broj sobe za koju se trenutno prate podaci u stvarnom vremenu. Nakon toga, kreiraju se tri dretve – prva dretva čita i upisuje podatke sobe koja se aktivno nadzire, druga čita i zapisuje podatke svih ostalih soba, dok treća dretva čita i zapisuje DHMZ podatke. Dretve izvršavaju čitanje i zapisivanje u intervalu zapisanom u bazi podataka koji je prethodno pročitan. Na kraju zapisivanja, ponovno se provjerava interval zapisan u bazi podataka te, u slučaju da je različit od onog koji je postavljen kod početka rada dretve, sve tri dretve se prekidaju. Nakon toga, s radom započinju tri nove dretve s novim intervalom. U slučaju promjene intervala usred zapisivanja u bazu podataka, zapisivanje će se uvijek dovršiti prije nego što dretva bude prekinuta.

Prethodno spomenuti podaci koji se koriste za simulaciju stanja pametnih soba zapisani su u CSV datotekama. Iz navedenih datoteka čita se redak po redak u zadanim intervalima, obrađuju se podaci te se naposljetku isti podaci nakon obrade spremaju u bazu. U Ispisu 5.1. vidljiv je programski kod koji služi za zapisivanje jednog retka iz CSV datoteke u bazu.

```
try {
                BufferedReader br = Files.newBufferedReader(Path.of(path),
StandardCharsets.UTF_8);
                WrappedReader wr = new WrappedReader(file, br, path);
                list.addAll(Reader.SingleLine(wr, ts, collectionName));
                Collection =
SmartRoomTrialDb.getCollection(IntervalController.priorityRoom);
                wr.getBr().close();
                wr = null;
            } catch (IOException ioe) {
            }
            try {
                assert Collection != null;
                Collection.insertMany(list, new InsertManyOptions().ordered(false));
            } catch (Exception e) {
                e.printStackTrace();
            }
```

*Ispis 5.1. Programski kod za upis jednog retka iz CSV datoteke u bazu*

Priloženi programski kod prikazuje unos podataka za prioritetnu sobu, odnosno sobu koja se aktivno nadzire,a taj kod izvršava prije spomenuta prioritetna dretva. Objekt `WrappedReader` služi kao razred omotač koji sadržava objekt `BufferedReader`, ime datoteke koju `BufferedReader` čita te stazu do te datoteke. Omotač razred napravljen je kako bi prijenos podataka između funkcija bio jednostavniji tako da se između funkcija šalje samo objekt `BufferedReader`, a ne svaka potrebna vrijednost zasebno. `BufferedReader` korišten je za čitanje podataka iz CSV datoteka. Samo čitanje izvodi se u funkciji `SingleLine` koja prima `WrappedReader` s podacima o datoteci, `Timestamp` koji će biti zapisan kao vrijeme očitanja te ime kolekcije za koju se čitaju podaci. Isječak koda za čitanje iz funkcije `SingleLine` prikazan je na Ispisu 5.2. `BufferedReader` pročita prvi redak datoteke, te se dobiveni `String` podijeli na vrijednosti razdvojene zarezom koje se potom unose u polje. Koristeći dobiveno polje stvara se instanca razreda `AdriaIndoorDataset`. Taj razred služi za preradu i privremeno spremanje podataka pročitanih iz CSV datoteka. Kod instanciranja objekta, podaci iz gore spomenutog polja obrađuju se u oblik prikladan za stvaranje `Document`-a. Naposljetku, briše se pročitani redak s vrha CSV datoteke.

```
            String s = br.getBr().readLine();

            String[] attributes = s.split(",");

            if (attributes.length != 0) {

                if(collectionName.equals(Filenames.AdriaCollectionName)){

                    s= s + "," + br.getFileName().substring(5 ,
br.getFileName().length()-4);

                    attributes = s.split(",");

                    AdriaIndoorDataset roomState = new AdriaIndoorDataset(attributes);

                    Document doc = AdriaIndoorDocument.createDoc(roomState , ts);

                    Warning.checkRoomStateAgainstRules(roomState , ts);

                    list.add(doc);

                }

            }

            StringManipulator.cleanFirstRow(br);
```

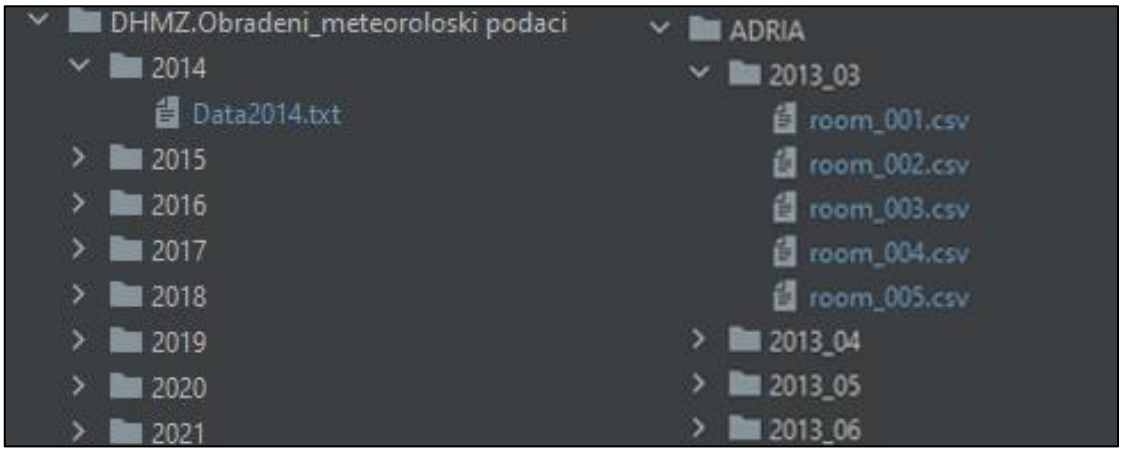
*Ispis 5.2. Programski kod za čitanje jednog retka iz CSV datoteke*

Od objekta dobivenog nakon čitanja podataka iz CSV datoteke stvara se `Document` koji će se unositi u bazu podataka. `Document` je objekt pomoću kojeg unosimo podatke u MongoDB bazu podataka koji radi na principu ključ - vrijednost te priložene podatke sprema u JSON formatu. Također, nužno je kao prvu vrijednost `Document`-a zadati primarni ključ, za što je korišten `ObjectId` koji garantira da će svi dokumenti imati različiti jedinstveni identifikator. Ispis 5.3. prikazuje primjer stvaranja `Document` objekta. Kod unosa podataka u stvarnom vremenu, za vrijeme se unosi trenutna vremenska značka (engl. *Timestamp*). Međutim, kod unosa arhivskih podataka koristi se vremenska značka zapisana u CSV datoteci.

```
public static Document createDoc(AdriaIndoorDataset data){

Document doc = new Document("_id" , new ObjectId());

         doc.append("vrijeme" , data.getTimestamp().getTime())

            .append("zadana" , data.getZadana())

            .append("izmjerena" ,data.getIzmjerena())

            .append("statusKlime" , data.getStatusKlime())

            .append("brzinaPuhanja" ,data.getBrzinaPuhanja())

            .append("ventil" , data.getVentil())

            .append("prisutnost" , data.getPrisutnost())

            .append("prozor" ,data.getProzor())

            .append("modKlime" , data.getModKlime())

            .append("wcSet" , data.getWcSet())

            .append("wcMjerenja" , data.getWcMjerena());

        return doc;

}
```

*Ispis 5.3. Programski kod za stvaranje dokumenta*

U sklopu aplikacije postoji i funkcija `ArchiveAllData` za unos arhivskih podataka. Spomenuta funkcija čita unaprijed pripremljene podatke iz CSV datoteka od najstarijih prema najnovijima te ih unosi u bazu podataka. Podaci moraju biti pripremljeni na određeni način kako bi se spremali u bazu na isti način kao što se spremaju i podaci u stvarnom vremenu. Struktura direktorija služi kao vodič za kolekcije u koje će se spremati dokumenti. Direktorij „*Adria*“ sadrži poddirektorije koji su imenovani na način „*godina_mjesec*“, pri čemu godina i mjesec predstavljaju vrijeme kada su izvršena očitanja. Unutar tih direktorija nalaze se odgovarajuće CSV datoteke s podacima iz pametnih soba koje su imenovane po brojevima soba u formatu „*room_brojSobe*“. Slično tome, DHMZ podaci sadržani su unutar direktorija „*Obradeni_meteoroloski_podaci*“ u kojem su direktoriji nazvani po godinama u kojima su očitanja nastala. Naposljetku, unutar tih direktorija nalaze se i CSV datoteke, također nazvane po godinama očitanja, u obliku „*Data2014*“. Na slici 5.1. prikazan je primjer takve strukture direktorija.




*Slika 5.1. Primjer strukture direktorija za unos arhivskih podataka*

Za DHMZ podatke potrebno je programski zadati popis godina za koje se unose podaci. Za podatke iz pametnih soba potrebno je zadati godine i mjesece iz kojih se unose podaci te brojeve soba iz kojih su podaci, to jest, imena CSV datoteka iz kojih se isti učitavaju. Za oba skupa podataka potrebno je zadati i stazu do samih podataka. Na kraju unosa arhivskih podataka, oni se indeksiraju po vremenu. Na Ispisu 5.4. prikazan je kod za indeksiranje arhivskih podataka.

```
for(String s : years){

            MongoDatabase db = mongoClient.getDatabase(s);

            for(String str : Filenames.adriaRoomNames){

                String c =  str.substring(5 , str.length()-4);

                String res =
db.getCollection(c).createIndex(Indexes.descending("vrijeme"));

               System.out.println("created index : "+ res + " in " + s + " " + str);

            }

            String res =
db.getCollection("DHMZObradeno").createIndex(Indexes.descending("vrijeme"));

            System.out.println("created index : "+ res +

            " in " + s + " " + "DHMZObradeno");

        }
```

*Ispis 5.4. Programski kod za indeksiranje arhivskih podataka*

# 6. Aplikacija za nadziranje pametnih soba

Aplikacija za nadzor pametnih soba sastoji se od dvije povezane cjeline – analitičke funkcije i funkcije nadzora u stvarnom vremenu. Cjeline su međusobno povezane tako da rade s istim podacima te im se može pristupiti s iste lokacije u aplikaciji. Na Slici 6.1. prikazana je naslovna stranica aplikacije za nadziranje.


*Slika 6.1. Prikaz naslovne stranice aplikacije za nadziranje*

Na naslovnoj stranici ispisane su instrukcije za korištenje aplikacije te kratka objašnjenja funkcija i način kako ih koristiti te kako im pristupiti. S naslovne stranice moguće je pristupiti bilo kojoj funkciji te provjeriti upozorenja.

## 6.1. Analitičke funkcije

Analitički dio aplikacije služi za analizu te promatranje podataka u arhivi. Moguće je prikazati sve podatke očitane u pametnim sobama te meteorološke prilike u različitim vremenskim rasponima. Pristupa im se kroz lijevu bočnu traku koja se otvara pritiskom na gumb „Brzi pristup“. Na bočnoj traci nalaze se 3 različite funkcije, što je vidljivo na Slici 6.2.


*Slika 6.2. Pristup funkcijama preko lijeve bočne trake*

Opcija „Stanje sob na dan“ vodi do funkcije prikaza stanja očitanih u pametnoj sobi na traženi dan. Pritiskom na tipku otvara se skočni prozor s poljima za unos podataka. Skočni prozor za ovu funkciju vidljiv je na Slici 6.3.

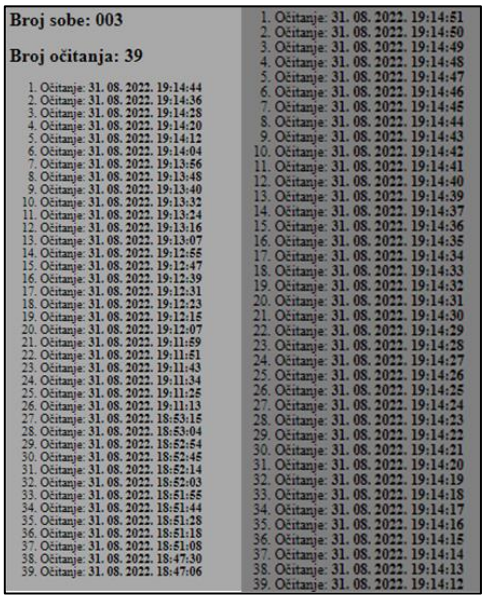

*Slika 6.3. Prikaz skočnog prozora za funkciju prikaza stanja sobe na traženi dan*

Korisnik unosi broj sobe koju želi provjeriti te datum za koji želi vidjeti očitana stanja. Nakon unosa podataka i pritiska zelene tipke, na lijevom ekranu se prikazuje popis svih očitanih stanja u toj sobi u rasponu od 24 sata, tj. na datum koji je zadan u skočnom prozoru. U slučaju velikog

broja očitanja, ona neće biti ispisana u cijelosti već će se prikazati poruka s brojem očitanja na taj dan. Primjer rezultata ove pretrage vidljiv je na Slici 6.4.


*Slika 6.4. Prikaz rezultata funkcije „stanje sobe na dan“*

Na slici 6.5. detaljnije su prikazani rezultati pretrage. S lijeve strane nalaze se očitanja pametne sobe, a s desne očitanja atmosferskih stanja.




*Slika 6.5. Detaljan prikaz poveznica na  pojedinačna očitanja*

Kroz sve funkcije aplikacije za nadzor, podaci iz pametnih soba i atmosferski podaci podijeljeni su u dvije sekcije. Na lijevoj strani uvijek su prikazani podaci iz pametnih soba dok su na desnoj uvijek podaci o atmosferskim stanjima. Ova podjela vrijedi za popise očitanja, prikaz očitanja na glavnom ekranu te grafove i posljednja očitanja na prozoru funkcije za nadzor u stvarnom vremenu. Sekcije se razlikuju po nijansi pozadinske boje.

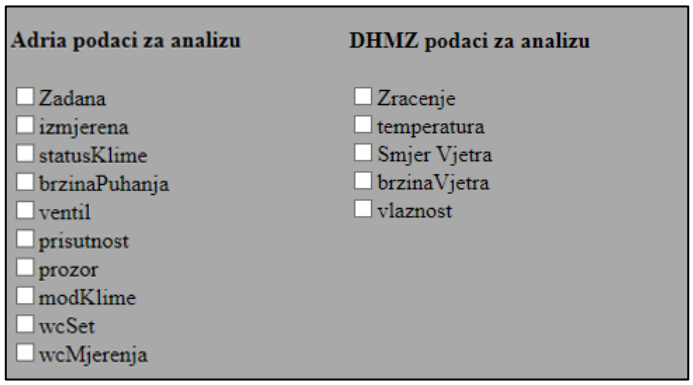
Pritiskom na jedno od očitanja otvara se skočni prozor s detaljnim prikazom svih izmjerenih vrijednosti u tom trenutku. Na Slici 6.6. vidljiv je skočni prozor s podacima.


*Slika 6.6. Prikaz očitanja u skočnom prozoru*

Radi bolje preglednosti podaci iz skočnog prozora mogu se prikazati na glavnom prozoru pritiskom na tipku „Prikaži na glavnom prozoru“. Prikaz podataka na glavnom prozoru vidljiv je na Slici 6.7.


*Slika 6.7. Prikaz podataka jednog očitanja na glavnom prozoru*

Osim pregleda pojedinačnih očitanja, moguće je grafički prikazati i uspoređivati različite vrijednosti iz očitanja soba i atmosferskih stanja. Ispod popisa očitanja nalaze se odabirni okviri za podatke iz očitanja pametnih soba te atmosferskih stanja, a pritiskom na odgovarajuću tipku moguće je prikazati sve odabrane vrijednosti na grafu radi vizualne usporedbe. Korisnik ima opciju usporediti podatke samo iz pametnih soba, samo atmosferske podatke ili oba skupa podataka na istom grafu. Na slici 6.8. moguće je vidjeti prikaz odabirnih okvira.

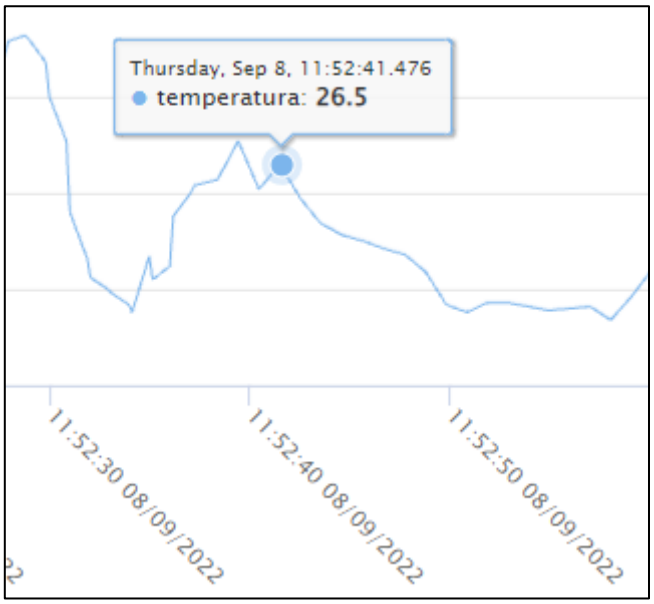



*Slika 6.8. Odabirni okviri za vizualnu analizu podataka*

Graf s odabranim podacima prikazuje se u skočnom prozoru te je na njemu moguće vidjeti vrijednost svakog pojedinačnog očitanja u vremenu. Primjer grafa s podacima o izmjerenoj temperaturi sobe i izmjerenoj atmosferskoj temperaturi vidljiv je na slici 6.9.


*Slika 6.9. Primjer grafa s atmosferskim podacima i podacima iz pametnih soba*

Graf je moguće uvećati kako bi se pobliže pregledali podaci na određenom intervalu tako da se mišem označi period na osi apscisi. Također, prelaskom miša iznad bilo kojeg djela grafa na x osi prikazuje se vrijednost varijable u tom trenutku unutar malog skočnog prozora, vidljivo na slici 6.10.

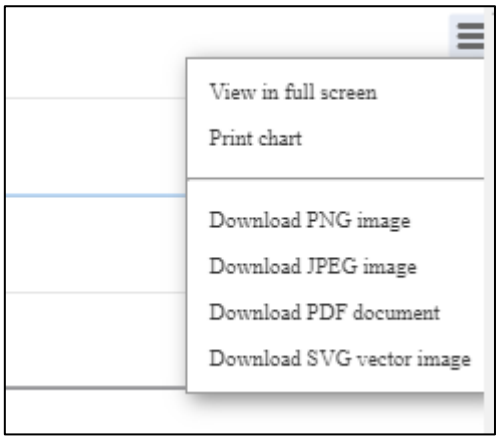



*Slika 6.10. Prikaz skočnog prozora s vrijednošću varijable*

Prilikom prikaza više različitih varijabli na grafu, moguće je privremeno isključiti prikaz pojedinačnih varijabli. Na dnu grafa nalaze se imena varijabli. Pritiskom na ime varijable uključuje se ili isključuje prikaz varijable na grafu. Na slici 6.11. prikazana su imena varijabli prikazanih na grafu s odgovarajućim bojama linija na grafu. Imena koja su izblijeđena označuju da je prikaz varijable isključen, dok su ona s punom bojom uključena.


*Slika 6.11. Prikaz imena varijabli prikazanih na grafu*

Pritiskom na tipku izbornika u desnom gornjem kutu grafa prikazane su još neke opcije koje su dane korisniku za upravljanje grafom. Dodatne opcije vidljive su na slici 6.12.




*Slika 6.12. Prikaz dodatnih opcija na grafu*

Graf je moguće otvoriti preko cijelog zaslona, poslati ga na ispis ili preuzeti grafički prikaz podataka u različitim formatima.

Sljedeća analitička funkcija je „Povijest sobe“. Pristupa joj se kroz opciju na lijevoj bočnoj traci pod imenom „Povijest sobe“. Funkcija služi za pregled arhivskih podataka u željenom vremenskom rasponu. Slično kao i funkcija prikaza stanja očitanih u pametnoj sobi na traženi dan, ova funkcija prikazuje sve podatke očitane u vremenskom periodu kojeg zadaje korisnik na skočnom prozoru. Kao i kod prijašnje funkcije, pristupa joj se preko skočnog prozora za unos podataka.

Korisnik mora unijeti broj sobe, datum od kojeg se traže podaci te datum do kojega se traže podaci. Skočni prozor vidljiv je na slici 6.13.

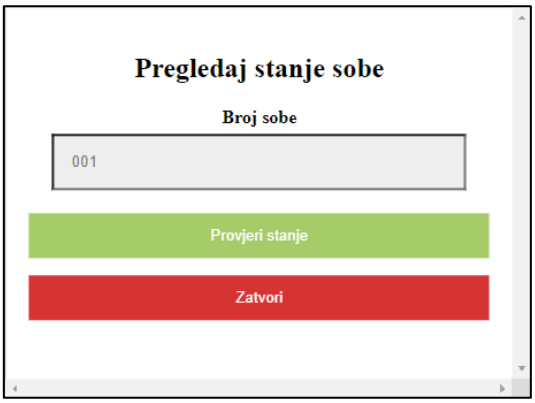

*Slika 6.13. Skočni prozor za pristup funkciji povijest sobe*

Pritiskom na tipku „Provjeri“ prikazuje se zaslon sličan onome od prijašnje funkcije. S obzirom na to da u rasponima većim od 24 sata postoji vrlo velika količina očitanja, za funkciju povijesti sobe nije dostupan popis očitanja. Ispod poruke o broju očitanja nalaze se izborni okviri s varijablama iz pametnih soba te varijablama atmosferskog stanja. Pomoću izbornih okvira odabiru se varijable koje se žele prikazati na grafu, na isti način kao i kod funkcije „Stanje sobe na dan“. Primjer korištenja funkcije „Povijest sobe“ vidljiv je na slici 6.14.


*Slika 6.14. Prikaz zaslona povijesti sobe*

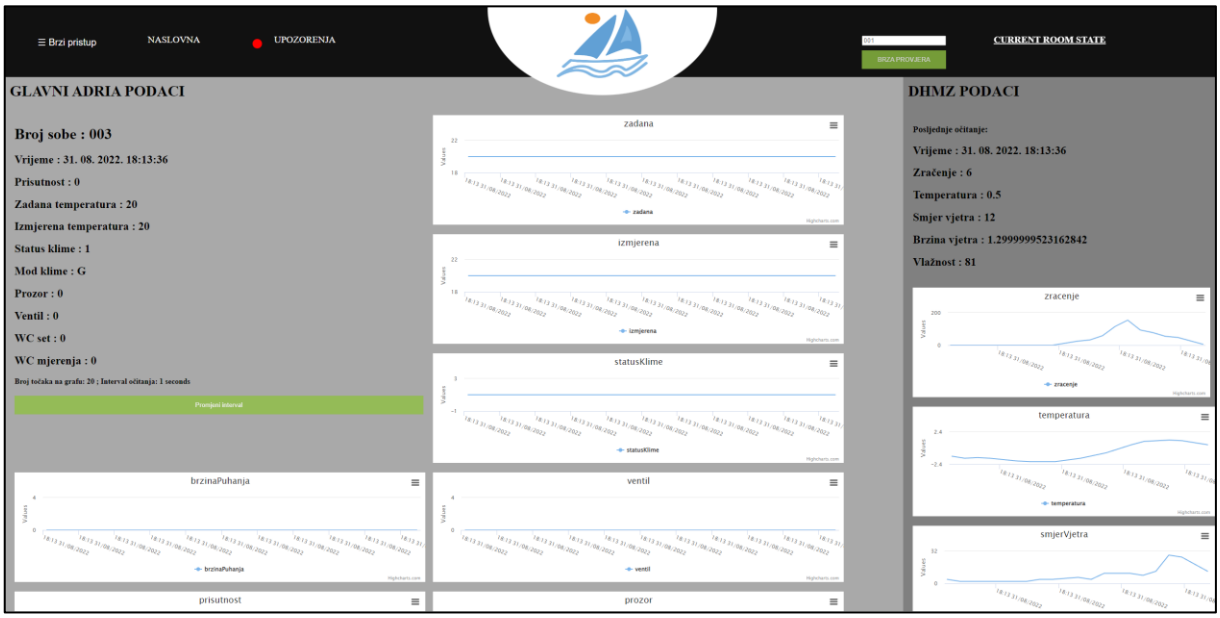
## 6.2. Funkcije nadzora u stvarnom vremenu

Glavna namjena aplikacije je nadzor stanja pametnih soba u stvarnom vremenu. Za nadzor u stvarnom vremenu postoje dvije funkcije – nadzor trenutnog stanja pojedinačnih soba te sustav za upozorenja. Do nadzora trenutnog stanja pojedinačnih soba dolazi se na tri različita načina. Prvi način je preko opcije „Stanje sobe“ koja se nalazi među opcijama na lijevoj bočnoj traci. Pritiskom na tipku otvara se skočni prozor u kojem se unosi broj sobe. Prikaz skočnog prozora vidljiv je na slici 6.15.




*Slika 6.15. Prikaz skočnog prozora za pristup nadzoru u stvarnom vremenu*

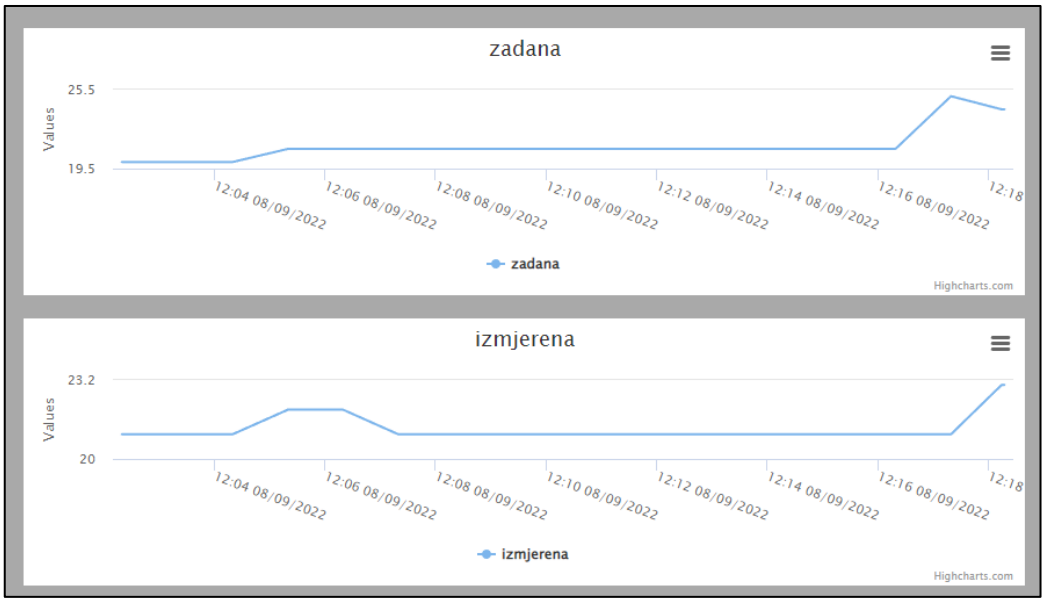
Drugi način za pristupiti istoj funkciji je putem opcije „Brza provjera“, koja se nalazi na glavnoj alatnoj traci, a za pokretanje aktivnog nadzora sobe potrebno je unijeti željeni broj sobe te pritisnuti tipku „Provjeri stanje“. U glavnom prozoru otvara se tekstualni prikaz posljednjih stanja uređaja očitanih u odabranoj pametnoj sobi, dok se s desne strane pojavljuju vrijednosti svih atmosferskih stanja prikupljenih prilikom posljednjeg očitanja. Prikaz aktivnog nadziranja pojedinačnih soba nalazi se na slici 6.16.




*Slika 6.16. Prikaz aktivnog nadziranja pojedinačne sobe*

Pored tekstualnog prikaza nalaze se grafovi s vizualnim prikazom svake varijable koja se ažurira u stvarnom vremenu. Grafovi su linijski te je broj točaka koje predstavljaju podskup najnovijih mjerenja ograničen. Inicijalno je ograničenje broja točaka na grafu postavljeno na deset, ali

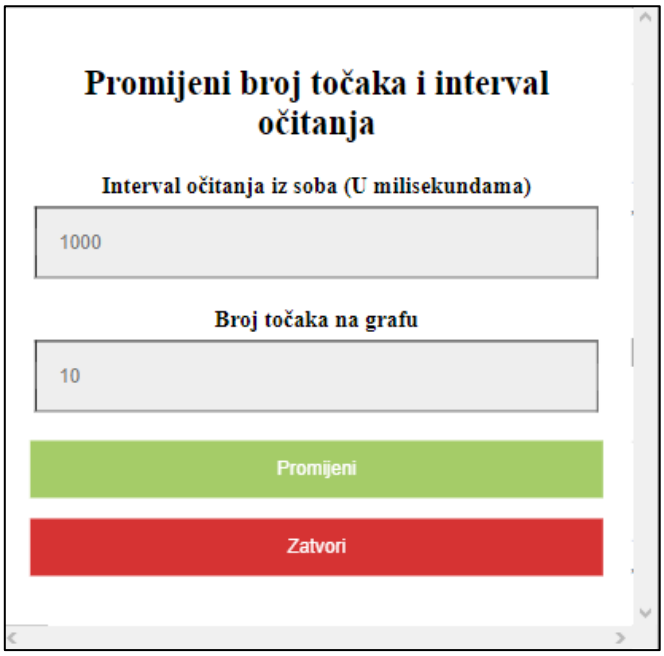
korisnik može promijeniti to ograničenje prilikom mijenjanja intervala očitanja. Na Slici 6.17. nalazi se detaljan prikaz dvaju grafova s podacima iz odabrane pametne sobe.




*Slika 6.17 Detaljan prikaz grafova*

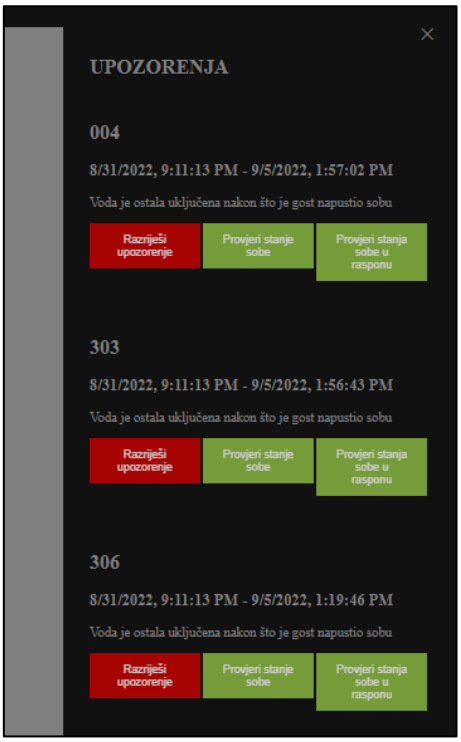
Grafovi ove funkcije sadrže sve mogućnosti kao i grafovi koji se crtaju unutar analitičkih funkcija.

Zadani interval očitanja je isti kao interval zadan prilikom zadnjeg aktivnog nadziranja neke sobe, to jest, čita se iz kolekcije „interval“ u bazi podataka. Ako u toj kolekciji ne postoji dokument, kreirat će se novi dokument s intervalom tisuću milisekundi te prioritetnom sobom koju je korisnik zadao. Interval očitanja i maksimalni broj točaka na grafu mogu se mijenjati pritiskom na tipku „Promijeni interval“, nakon čega se otvara novi skočni prozor u kojem je moguće unijeti nove željene vrijednosti. U prvo polje unosi se željeni interval očitanja izražen u milisekundama. U drugo polje unosi se željeni broj točaka koje su odjednom prikazane na grafu. Pritiskom na tipku „Promijeni“ aplikacija će se ponovo učitati s novim intervalom. Na Slici 6.18. vidljiv je prikaz skočnog prozora za promjenu intervala.



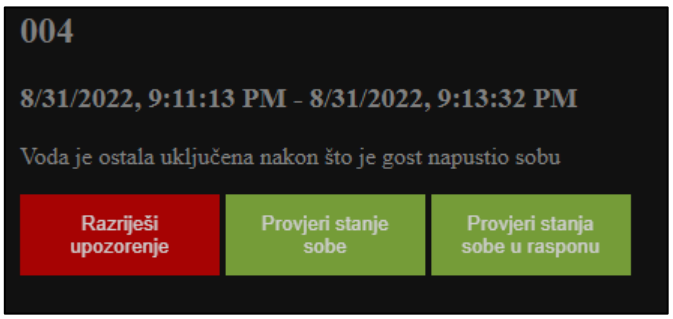

*Slika 6.18. Prikaz skočnog prozora za promjenu intervala očitanja*

Pri svakom očitanju podataka iz pametnih soba, podaci se provjeravaju za predodređena zabranjena stanja. U slučaju detekcije zabranjenog stanja u nekoj sobi, u bazu podataka se zapisuje dokument s informacijama o tom očitanju. Svakih deset sekundi aplikacija provjerava stanja zapisana u bazi podataka te obavještava korisnika o mogućim zabranjenim stanjima. Ako se u bazi podataka pronađu zapisana upozorenja, korisniku će to biti naznačeno svjetlećom crvenom točkom kraj tipke za upozorenja. Upozorenja se prikazuju u desnoj bočnoj traci koja se otvara pritiskom na tipku „Upozorenja“. Na slici 6.19. vidljiv je prikaz upozorenja u otvorenoj desnoj bočnoj traci.




*Slika 6.19. Prikaz desne bočne trake s upozorenjima*

Upozorenje se sastoji od broja sobe, datuma i vremena kada je prvi puta primijećeno određeno odstupanje, zatim datum i vrijeme zadnjeg odstupanja, poruke koja opisuje dotični problem te 3 tipke za provjeru i razrješavanje upozorenja. Na desnoj bočnoj traci za upozorenja nisu prikazana upozorenja za pojedinačna očitanja kod kojih je detektiran problem, kao što je zapisano u bazi podataka, već su ta upozorenja grupirana prema pravilu koje je prekršeno. Tako se izbjegava nepreglednost jer svaka soba može imati samo jedno upozorenje za svaki tip prekršaja. Grupiranje upozorenja bit će detaljno opisano u sljedećim poglavljima. Slika 6.20. detaljnije prikazuje strukturu upozorenja.




*Slika 6.20. Prikaz upozorenja*

Korisniku su dane 3 opcije za svako upozorenje:

1. Provjeri stanje sobe
2. Provjeri stanja sobe u rasponu
3. Razriješi upozorenje

Funkcija „Provjeri stanje sobe“ je treći način kako korisnik može doći do funkcije za nadzor u stvarnom vremenu. Pritiskom na tu tipku korisnik se upućuje direktno na funkciju za nadzor sobe za koju je izdana odabrana obavijest, bez potrebe da se otvara skočni prozor ili unose ikakvi dodatni podaci.

Funkcija „Provjeri stanja sobe u rasponu“ služi kao poveznica između analitičkog djela aplikacije i djela za nadzor u stvarnom vremenu. Pritiskom na tipku „Provjeri stanja sobe u rasponu“, poziva se funkcija „Povijest sobe*“* za prikaz svih očitanja te sobe u vremenskom rasponu unutar kojeg je primijećeno zabranjeno stanje.

Nakon provedenih provjera za upozorenje, korisnik ima opciju razriješiti potencijalna upozorenja koja su se pojavila. Pritiskom na tipku „Razriješi upozorenje“, upozorenje će biti obrisano iz korisničkog sučelja. Iz baze podataka bit će obrisana sva upozorenja tog tipa za dotičnu sobu. Bitno je napomenuti da brisanje upozorenja neće obrisati očitanja na kojima je primijećeno odstupanje.

# 7. Implementacijski detalji od posebnog značaja

Prilikom razvoja aplikacije za nadzor, kao i aplikacije za simulaciju pametnih soba, bilo je potrebno donijeti brojne odluke o implementaciji baze podataka i načinu čitanja i zapisivanja podataka u bazu podataka kako bi aplikacija za nadzor i simulacijska aplikacija zajedno radile optimalno. U ovom poglavlju istaknute su i pobliže opisane dvije značajnije odluke kod implementacije aplikacija – praćenje podataka u stvarnom vremenu i prikaz upozorenja u stvarnom vremenu.

## 7.1 Praćenje podataka u stvarnom vremenu

Zbog potrebe nadzora pametnih soba u stvarnom vremenu, vrijeme zapisivanja i čitanja iz baze podataka moralo se svesti na što manji period koji korisnik može slobodno promijeniti u svakom trenutku. Kako bi se postigla takva fleksibilnost, korištene su različite dretve za unos različitih podataka u bazu podataka. Prilikom pokretanja simulatora učitava se interval i broj sobe koja se trenutno nadzire iz kolekcije „interval“ te se s tim podacima stvaraju dretve.

S obzirom na to da se u isto vrijeme očitava stanje vrlo velikog broja soba, zapisivanje svakog očitanja u bazu podataka za sve sobe iznosi otprilike sedam do osam sekundi. To nije dovoljno brzo jer stanje u sobama želimo moći pratiti iz sekunde u sekundu. Kako bi podatke za sobu koja se trenutno aktivno nadzire dobivali u manjim intervalima, na razini simulatora napravljene su 3 dretve. Dvije dretve služe za unos DHMZ podataka i podataka svih soba, dok zadnja dretva prima podatke o sobi koja se trenutno aktivno nadzire te unosi u bazu samo očitanje te sobe. Tako je omogućeno prioritiziranje sobe koju aktivno nadziremo te podatke iz te sobe možemo dohvaćati u intervalima manjima od jedne sekunde. Tijekom mijenjanja intervala s korisničke strane, novi interval te broj sobe koja se aktivno nadzire zapisuju se u bazu podataka u kolekciju „interval“. Dretve na kraju zapisa podataka provjeravaju interval i broj sobe koji su zapisani u bazi podataka te, u slučaju bilo kakve promjene, prekidaju s radom te započinju novi ciklus s aktualnim parametrima. U nastavku je prikazan Ispis 7.1., na kojem je moguće vidjeti programski kod za inicijalizaciju i pokretanje svih spomenutih dretvi.

Pri kraju svakog čitanja i zapisa u bazu, pokrenuta je naredba `thread.sleep()`. Dretva ostaje u tom stanju minimalno sto milisekundi, a maksimalno do kraja početno zadanog intervala čitanja.

Dretva se stavlja u stanje *sleep* kako bi dobili priliku prekinuti je u slučaju da korisnik promijeni interval očitanja ili broj sobe koja se aktivno nadzire.

```
try {
            for (Thread t : threads) {
                System.out.println(t.getName() + " Trying to close");
                t.interrupt();
            }
            priorityRoom = roomId;
            intervalGlobal = interval;
            Thread t0 = new Simulator(Filenames.AdriaCollectionName, interval, true);
            Thread t1 = new Simulator(Filenames.AdriaCollectionName, interval , false);
            Thread t2 = new Simulator(Filenames.DHMZObradenoCollectionName,
                        Interval, false);
            threads.add(t0);
            threads.add(t1);
            threads.add(t2);
            t0.start();
            t1.start();
            t2.start();
        } catch (Exception e) {
            System.out.println(e.getStackTrace());
            return e.getMessage();
        }
        return "Success, changed interval to: " + interval;
    }
```

*Ispis 7.1. Prikaz programskog koda koji započinje dretve*

### 7.1. Upozorenja u stvarnom vremenu

Prilikom nadgledanja pametnih soba, potrebno je obavijestiti korisnika o mogućim zabranjenim stanjima. Kako bi se smanjio utjecaj ovih provjera na brzinu čitanja i pisanja u bazu podataka, napisana su određena pravila koja se provjeravaju prilikom unosa podataka u bazu. Primjer jednog od njih je pravilo „Prozor je otvoren i klima je upaljena dok gost nije prisutan u sobi“. Kod upisa u bazu podataka, stanje sobe se provjerava s obzirom na sva zadana pravila. U slučaju kršenja pravila, u bazu se zapisuje dokument s brojem sobe u kojoj je detektiran prekršaj, vremenom u UNIX formatu te brojem pravila koje je prekršeno. Na ispisu 7.2. vidljiv je primjer provjere očitanja za kršenje pravila.

```
public static void checkRoomStateAgainstRuleOne(AdriaIndoorDataset roomState ,
                    Timestamp ts) {
        if (roomState.getPrisutnost() != 1 && roomState.getStatusKlime() != 0 &&
            roomState.getProzor() != 0) {
            MongoClient mongoClient = Connect.getClient().mongoClient;
            MongoDatabase SmartRoomTrialDb = mongoClient.getDatabase("2022");
            MongoCollection<Document> Collection =
            SmartRoomTrialDb.getCollection("Warnings");
            Collection.insertOne(WarningDocument.createDoc(roomState.getRoomName(),
                                rule1, ts , 1));
        }
    }
```

*Ispis 7.2. Programski kod za provjeru očitanja za kršenje pravila*

S korisničke strane radi se *HTTP* zahtjev za upozorenjima. Primitkom upita na poslužiteljskoj strani čitaju se podaci iz kolekcije „Warnings“ u bazi podataka. Dobiveni podaci zatim se grupiraju prema broju sobe i pravilu koje je prekršeno. Tako se osigurava da svaka soba odjednom može imati samo jedno upozorenje za svako prekršeno pravilo. Prilikom grupiranja, zapisuje se vrijeme prvog očitanja kod kojeg je detektiran prekršaj te zadnje očitanje s prekršajem. Na ispisu 7.3. vidljiv je programski kod za grupiranje upozorenja.

Unutar intervala koji nastaje grupiranjem upozorenja na ovaj način mogu se pronaći i očitanja koja nisu u stanju prekršaja. Do toga dolazi ako neki podatak prekrši zadano pravilo, vrati se u dozvoljeno stanje te nakon toga ponovo prekrši isto pravilo.

Grupiranje na ovaj način je potrebno zbog mogućih brzih izmjena stanja iz dopuštenog u nedopušteno. Do toga može doći zbog neispravnih uređaja. Primjerice, uređaj za mjerenje temperature zbog neke neispravnosti mogao bi povremeno detektirati pogrešnu temperaturu, a povremeno raditi ispravno. Kada se ta upozorenja ne bi grupirala na ovaj način, bočna traka koja prikazuje upozorenja bi se vrlo brzo mogla napuniti velikim brojem upozorenja iste vrste za istu sobu, što bi moglo aplikaciju učiniti neresponzivnom te samo korištenje sustava za upozorenja nepreglednim.

```
for (Warning item : handled) {
                    if (item.getRoomName().equals(war.getRoomName()) &&
                        item.getRuleNo() == war.getRuleNo()) {
                        if (item.getTimestampFrom() < war.getTimestampFrom()) {
                            item.setTimestampTo(war.getTimestampFrom());
                            tmp = null;
                            break;
                        } else if (item.getTimestampFrom() > war.getTimestampFrom()) {
                            item.setTimestampTo(item.getTimestampFrom());
                            item.setTimestampFrom(war.getTimestampFrom());
                            tmp = null;
                            break;
                        }
                    }
                    tmp = war;
                }
                if (tmp != null) handled.add(tmp);
                tmp = null;
```

*Ispis 7.3. Programski kod za grupiranje upozorenja*

# 8. Zaključak

Dostupnost različitih vrsta informacija u svakom trenutku pokazalo se od iznimne važnosti u današnjem svijetu, što se uvelike odražava i u hotelijerskoj industriji. Aplikacija za nadzor pametnih soba ciljanom korisniku na bilo kojoj lokaciji u bilo koje doba dana, omogućuje dostupnost relevantnih informacija u stvarnom vremenu. Uz nadzor podataka, obrada podataka u stvarnom vremenu pruža i mogućnost analize očitanih podataka. Takvo što je jako korisno u slučajevima poput servisiranja uređaja, planiranja nadogradnji pametnih soba te nadzor aktivnosti unutar hotela.

Zbog velikih mogućih količina podataka koji se obrađuju unutar analitičkih funkcija, postoje određena ograničenja pri implementaciji i korištenju ovakve vrste aplikacija. Kod pregleda očitanja u različitim periodima, količina podataka može prikaz svih pojedinačnih očitanja učiniti vrlo nepreglednim. Stoga se ograničava broj očitanja koja mogu biti pojedinačno prikazana kod pretrage u vremenskom rasponu. Također, u slučaju velikih vremenskih raspona, dohvaćanje podataka za grafičku analizu i samo crtanje grafova postaje značajno vremenski duže. Slučaj grafičke vizualizacije svih podataka iz pametnih soba, kao i atmosferskih podataka u rasponu od 2013. do 2021. godine, na istom grafu traje do 90 sekundi. Ovaj slučaj nije ciljani kontekst korištenja aplikacije, već je uzet kao onaj rubni, tj. najzahtjevniji slučaj za testiranje mogućnosti sustava. Podrazumijeva dohvaćanje oko 150 milijuna dokumenata iz baze podataka te prikaz tih podataka na grafu, što rezultira s oko $2e^9$ točaka na grafu. Kod ovako velikih skupova podataka, graf postaje slabije responzivan te nepregledan.

Kod praćenja podataka u stvarnom vremenu moguće je interval očitanja postaviti na vrijednosti manje od sekunde. Međutim, kod intervala manjih od sekunde dolazi do manjih nekonzistentnosti u vremenima između očitanja. S obzirom na to da je baza podataka sadržana u potpunosti u oblaku, kod svakog očitanja potrebno je vrijeme da podatak bude poslan sa simulatora u bazu podataka te zatim iz baze podataka na korisničku stranu. Dodatno vrijeme za slanje i dohvaćanje podataka ovisi o trenutnoj internetskoj vezi kao i poslužitelju na kojem je sadržana baza podataka. Slično kao i s nedostatkom analitičkih funkcija, ovaj nedostatak nije ciljani kontekst korištenja aplikacije, već je uzet kao rubni slučaj. Čak i kod nekonzistentnog vremena dohvaćanja podataka, rezultiranog premalim intervalom očitanja, aplikacija na korisničkoj strani podatke prikazuje ispravno te ovo ograničenje ne utječe odveć na korisnikov doživljaj.

Kao nadogradnja implementiranog sustava predlažu se moguće prilagodbe korisničkog sučelja prema željama i potrebama korisnika i objekta koji se nadzire. U slučaju manjeg broja soba, sustav

za upozorenja mogao bi se napraviti uočljivijim. Primjerice, početna stranica mogla bi se sastojati od tipki s brojevima soba koje bi mijenjale boju ovisno o određenim prekršajima ili upozorenjima te bi vodile korisnika direktno do funkcije za aktivno nadziranje sobe. Također, moglo bi se dodati više načina vizualizacije podataka osim linijskih grafova. Nadalje, mogu se dodati nove provjere za slanje upozorenja kod detektiranja novih prekršaja.

# 9. Bibliografija


[1] MongoDB : „MongoDB documentation“, s interneta, https://www.mongodb.com/docs/ , 30.6.2022.

[2] Phillip Webb, Dave Syer, Josh Long i dr.: „Spring Boot reference Documentation“, s interneta, https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/, 30.6.2022.

[3] Codeacademy Team: „What is REST?“, s interneta, https://www.codecademy.com/article/what-is-rest 30.6.2022.

[4] Axios : „Getting Started | Axios docs“, s interneta, https://axios-http.com/docs/intro , 30.6.2022.

[5] Highcharts : „Highcharts Documentation“, s interneta, https://www.highcharts.com/docs/index, 30.6.2022.

[6] Državni hirdometeorološki zavod : „Zahtjevi za podacima i uslugama “, s interneta, https://meteo.hr/proizvodi.php?section=katalog_zahtjevi&param=zahtjev_podaci_usluge, 5.9.2022.

# SAŽETAK

U ovome radu predstavljena je web aplikacija sa sučeljem prema bazi podataka koja sadrži informacije iz sustava pametnih soba. Osim mjernih podataka s različitih osjetila smještenih unutar pametnih soba dotičnog hotela, baza dodatno sadrži i meteorološke podatke za odgovarajuće vrijeme i fizičku lokaciju. Razvijena aplikacija simulira prihvat navedenih podataka, a odgovarajućom vizualizacijom omogućava kako praćenje stanja u pametnim sobama u stvarnom vremenu, tako i analizu stanja u prošlosti. Krajnjem korisniku na raspolaganju su funkcije koje podržavaju analizu podataka u željenim vremenskim rasponima, pregled pojedinačnih očitanja iz odabranih pametnih soba, aktivno praćenje stanja u stvarnom vremenu te upravljanje upozorenjima u slučaju detekcije anomalija.



***Ključne riječi — pametne sobe, nadzor u stvarnom vremenu, vizualizacija podataka, MongoDB***



# ABSTRACT

In this thesis, a web application with an interface to a database that contains information from the smart room system is presented. In addition to measurement data from various sensors located inside the smart rooms of the respective hotel, the database also contains meteorological data for the corresponding time and physical location. The developed application simulates the inflow of the mentioned data, and with appropriate visualization, it enables smart room monitoring in real-time and analysis of past conditions. The end user is provided with functions that support data analysis within the particular time ranges, measurements inspection for selected smart rooms, active monitoring in real-time, and warning management in case of anomaly detection.



***Keywords — smart rooms, real-time monitoring, data visualization, MongoDB***