Upravljačka ploča za mrežu mjerača kvalitete zraka

Klarić, Katarina

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1262

UPRAVLJAČKA PLOČA ZA MREŽU MJERAČA KVALITETE ZRAKA

Katarina Klarić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1262

UPRAVLJAČKA PLOČA ZA MREŽU MJERAČA KVALITETE ZRAKA

Katarina Klarić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1262

Pristupnica:	**Katarina Klarić (0036539648)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	izv. prof. dr. sc. Ana Babić

Zadatak: **Upravljačka ploča za mrežu mjerača kvalitete zraka**

Opis zadatka:

Ugljični dioksid je jedan od najvažnijih parametara kvalitete zraka u zatvorenim prostorima. Povišena koncentracija može negativno utjecati na zdravlje i usredotočenost, a pokazano je i da je dobar indikator koncentracije aerosola. Zadatak je izraditi mrežnu aplikaciju s upravljačkom pločom (dashboard) koja omogućuje praćenje kvalitete zraka na više mjernih lokacija. Aplikacija treba dati brzi pregled statusa uređaja i zadnjih očitanih podataka. Svaki od prikaza treba biti interaktivan te omogućiti detaljniji pregled i analizu. Aplikacija mora imati javno dostupan pregled podataka te provjeru identiteta korisnika kojem je dozvoljeno unošenje vrijednosti i komentara u bazu podataka.

Rok za predaju rada: 14. lipnja 2024.

Sadržaj

Uvod 1

1. Kvaliteta zraka u zatvorenim prostorima 2

1.1. Najčešći onečišćivači zraka 2

1.2. Zakonske regulative 4

1.3. Mjerni uređaji 5

2. Izrada aplikacije 8

2.1. Funkcionalnost aplikacije 8

2.1.1. Upravljačka ploča 8

2.1.2. Baza podataka 10

2.2. Korištene tehnologije i alati 11

2.2.1. Radni okvir Dash 11

2.3. Aplikacija 14

2.3.1. Upravljačka ploča 14

2.3.2. Baza podataka 22

2.3.3. Autentifikacija 26

2.4. Ograničenja i poboljšanja 27

Zaključak 29

Literatura 30

Sažetak 32

Summary 33

Uvod

Europski državljani u prosjeku provode 90% svog života u zatvorenim prostorima. Za razliku od vanjskog zraka, zrak u zatvorenim prostorima ne podliježe velikom broju zakonskih odredbi koje se odnose na njegovu kvalitetu. Na kvalitetu zraka u zatvorenim prostorima utječu brojni zagađivači poput hlapivih organskih spojeva, ugljikovog monoksida te lebdećih čestica $PM_{2.5}$ i PM_{10}. Oni mogu imati negativan utjecaj na ljudsko zdravlje, a neki od njih su i kancerogeni. Ugljikov dioksid, osim u vrlo visokim koncentracijama, nije štetan za zdravlje, ali može izazvati osjećaj pospanosti, glavobolju, smanjiti fokus i produktivnost. Koncentracija ugljikovog dioksida u zatvorenim prostorima pokazuje koliko je dobra ventilacija, a time i kvaliteta zraka u prostoru.

Cilj ovog rada bila je izrada mrežne aplikacije za vizualizaciju mjerenja dobivenih od mreže mjerača kvalitete zraka i interaktivan prikaz zadnjih očitanja koncentracije ugljikovog dioksida na više lokacija. Također je trebalo omogućiti prikaz najnovijih podataka iz baze i njihovu izmjenu autoriziranim korisnicima. Za implementaciju programskog rješenja korišten je radni okvir Dash i programski jezik Python. Dash omogućava brzu i jednostavnu izradu interaktivnih aplikacija za prikaz analitičkih podataka.

U ovom radu opisat će se važnost mjerenja kvalitete zraka u zatvorenim prostorima i glavni zagađivači zraka s naglaskom na ugljikov dioksid. Zatim će se objasniti način izrade i korištenja upravljačke ploče te opisati pojedini dijelovi koda. Na kraju će se navesti ograničenja i moguća poboljšanja rada aplikacije.

1. Kvaliteta zraka u zatvorenim prostorima

Na kvalitetu zraka u zatvorenom prostoru utječu spojevi koji se oslobađaju isparavanjem građevinskih materijala i premaza s namještaja, parfemi te tvari koje se oslobađaju znojenjem, čišćenjem, korištenjem printera i kuhanjem [1]. Ljudi su jedan od glavnih izvora zagađivača. Disanjem se oslobađa ugljikov dioksid u zrak, isparavanjem vlage iz kože oslobađaju se neugodni mirisi, a ljuštenje kože i ispadanje kose doprinose stvaranju prašine. Respiratorne čestice koje su raspršene u zraku nazivaju se aerosoli. Virusi i drugi zagađivači mogu živjeti na aerosolima [2]. Aerosoli se raspršuju kad kašljemo, govorimo i dišemo te je zbog toga bitno imati dobru ventilaciju u zatvorenim prostorima. Pribor za pisanje, papir i električni uređaji tijekom korištenja otpuštaju razne organske spojeve u zrak. Sastojci sredstava za čišćenje isparavaju pri korištenju i mogu zagaditi unutrašnji zrak na dulje vrijeme. Najznačajniji izvor zagađivača u zatvorenim prostorima u Europi dugo je bio duhanski dim, no danas je u većini zatvorenih prostora pušenje zabranjeno. Vanjski zagađivači poput ispušnih plinova automobila, otpadnih plinova iz industrijskih postrojenja, agrikulturnih proizvoda, peludi i spora ulaze u prostorije kroz ventilacijski sustav ili prozore.

1.1. Najčešći onečišćivači zraka

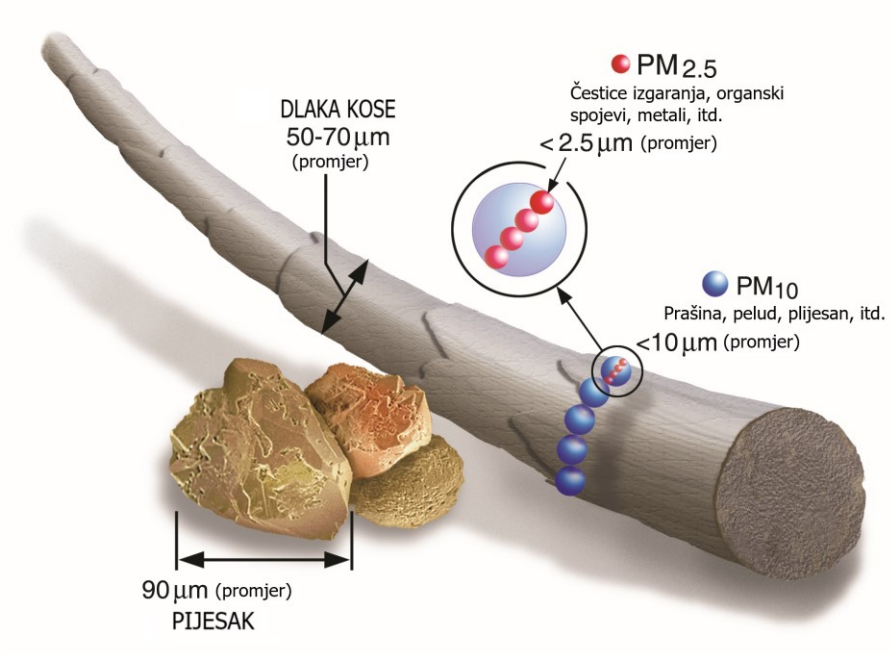
Ugljikov monoksid plin je bez mirisa, boje i okusa. Nastaje nepotpunim izgaranjem organskih spojeva. Otrovan je jer smanjuje sposobnost krvi za prijenos kisika. U zatvorenim prostorima nastaje kao posljedica uporabe neispravnih uređaja za grijanje ili kuhanje koji rade na plin, ugljen i drvo. WHO preporučuje vrijednost od 10 mg/m³ u zatvorenim prostorima unutar 8 sati [1].

Hlapivi organski spojevi (engl. *Volatile organic compounds*, skraćeno VOC) su ugljikovodici koji lako hlape na sobnoj temperaturi i mogu imati štetne posljedice na ljudsko zdravlje. Neke od njih su iritacija očiju, nosa i grla, respiratorni problemi, alergijske reakcije na koži, vrtoglavica, mučnina, glavobolje, ali mogu imati i kancerogeni učinak. Potencijalni izvori VOC-a su boje i lakovi, sredstva za čišćenje, sredstva za zaštitu bilja, materijali za graditeljstvo, foto-kopirni uređaji, printeri, trajni markeri te razna ljepila [3]. Koncentracija VOC-a računa se tako da se zbroji koncentracija pojedinačnih spojeva

pod nazivom TVOC. Europska komisija objavila je najvišu dozvoljenu koncentraciju od 300 $\mu g/m^3$ TVOC-a u zatvorenim prostorima [1].

Formaldehid je kancerogen bezbojan plin oštra mirisa koji se također može pronaći u zatvorenim prostorima kao posljedica otpuštanja materijala za gradnju i opremu prostora te korištenjem sredstava za čišćenje i dezinfekciju [30]. Nalazi se u bojama i lakovima, namještaju, tekstilima i dezinficijensima. WHO preporučuje maksimalnu vrijednost od 0.1 mg/m^3 za 30 minutnu izloženost [1]. **Radon** je inertan plin bez boje, mirisa i okusa koji se također može naći u zatvorenim prostorima. Nastaje radioaktivnim raspadom uranija i torija koji se nalaze u tlu. Uzrokuje rak pluća.

Lebdeće čestice ili PM (engl. *Particulate Matter*) smjesa su čvrstih i tekućih čestica raspršenih u zraku te se sastoje od raznih kemijskih spojeva. PM_{10} lebdeće čestice imaju promjer 10 mikrometara ili manji, a $PM_{2.5}$ su čestice promjera 2.5 mikrometara ili manje. Ljudska dlaka ima promjer 70 mikrometara što znači da je 30 puta veća od $PM_{2.5}$ čestica [4]. Usporedba veličine dlake kose i lebdećih čestica može se vidjeti na slici 1.1. Ove čestice mogu ući u pluća, a manje čak i u krvotok te uzrokovati ozbiljne zdravstvene probleme. WHO propisuje graničnu vrijednost od 25 $\mu g/m^3$ $PM_{2.5}$ kroz 24 sata, a 50 $\mu g/m^3$ za PM_{10} [1].



Slika 1.1 Usporedba veličine lebdećih čestica i dlake kose [4]

Ljudi sami ispuštaju mikroorganizme u okoliš prilikom izdisanja ili perutanjem kože. **Mikroorganizmi** u kombinaciji s vlagom i nedovoljnom ventilacijom onečišćuju zrak u

zatvorenim prostorima. Vlaga uzrokuje rast plijesni, gljivica i bakterija na gotovo svim unutrašnjim materijalima. Kao rezultat, VOC se otpuštaju u većim količinama. Europska komisija propisala je količinu iznad 500 CFU/m^3 (CFU – jedinica formiranja kolonija) kao srednje zagađenje, a količinu iznad 2000 CFU/m^3 kao visoko zagađenje mikroorganizmima u zatvorenim prostorima [1].

Ugljikov dioksid plin je bez boje i mirisa te je prirodni sastojak zraka s prosječnom koncentracijom od 400 ppm. Ljudsko disanje glavni je izvor ugljikovog dioksida u zatvorenim prostorima. Količina ugljikovog dioksida u zraku izražava se kao ppm (engl. *parts per million*) – dijelova na milijun. Zrak na otvorenom sadrži otprilike 380 ppm ugljikovog dioksida osim u prometnim područjima i u blizini industrijskih pogona gdje je njegova koncentracija veća. U zatvorenim prostorima koncentracija CO_2 je veća prvenstveno zbog ljudi koji u njima borave. Jedan izdah prosječne odrasle osobe sadrži između 35000 i 50000 ppm ugljikovog dioksida [5], stoga je bitno uspostaviti odgovarajuću ventilaciju koja će ga ukloniti. U školama i uredima koncentracija CO_2 je uglavnom daleko ispod 5000 ppm što predstavlja sigurnosnu granicu. Koncentracije ispod 5000 ppm ne predstavljaju ozbiljniju zdravstvenu prijetnju, ali se pojedinci žale na osjećaj pospanosti, umora i glavobolje što može utjecati na njihovu produktivnost [5]. Povišena razina CO_2 u zatvorenim prostorima ukazuje na nedovoljnu ventilaciju vanjskim zrakom i lošu kvalitetu zraka s mogućim navedenim opasnijim zagađivačima.

1.2. Zakonske regulative

Uspostavljeni su mnogi standardi i smjernice za koncentraciju CO_2 u stambenim i poslovnim zgradama u različitim državama. U SAD-u je prihvaćen standard za ventilaciju koji propisuje ASHRAE (*American Society of Heating, Refrigeration, and Air Conditioning Engineers*) pod nazivom „Standard 62“. U skladu s tim standardom učionice trebaju biti opskrbljene s 0.43 m^3/min (15 cfm - kubična stopa po minuti) vanjskog zraka po osobi, a uredi s 0.57 m^3/min (20 cfm). Preporučuje da koncentracija CO_2 u školama ne bude iznad 1000 ppm, a u uredima iznad 800 ppm [5], ali naravno te brojke ovise o koncentraciji CO_2 u vanjskom zraku. Koncentracija CO_2 u zatvorenim prostorima ne bi smjela prekoračiti okolnu koncentraciju za više od 700 ppm [6]. Većina uspostavljenih standarda u različitim državama preporučuje održavanje razine CO_2 ispod 1000 ppm.

U Nizozemskoj, Austriji, Francuskoj, Norveškoj, Finskoj, Danskoj i Njemačkoj postoje standardi koji propisuju prihvatljivu razinu CO_2 u zatvorenim prostorima, kao što su škole, stambene i poslovne zgrade, u vrijednosti do 1000 ppm, a ponegdje 1500 do 2000 ppm kao gornju granicu. Standardi u Japanu, Južnoj Koreji i Novom Zelandu također navode 1000 ppm kao graničnu vrijednost [6].

U Belgiji je donesen zakon (Zakon 6. studenog 2022.) čija je svrha poboljšanje kvalitete zraka u zatvorenim prostorima dostupnima javnosti [7]. Taj zakon daje preporuke za dobru kvalitetu zraka, ali ne obvezuje odgovorne za rukovođenje tim prostorima da ih se pridržavaju. Te preporuke omogućavaju procjenjivanje trenutne kvalitete zraka u prostoru. Zakon potiče one odgovorne za te prostore da unaprijede kvalitetu zraka tako da je mjere s CO_2 mjeračima, naprave popis mogućih zagađivača zraka u njihovim prostorima i opišu korištene ventilacijske sustave te naprave analizu rizika. Također se preporuča da naprave akcijski plan u kojem će opisati načine na koje planiraju poboljšati kvalitetu zraka i dovesti je do referentnih razina. Zakon osigurava određenu razinu transparentnosti s obzirom na kvalitetu zraka koja se može očekivati u navedenim zatvorenim prostorima kroz uvođenje certifikata i oznaka. Svaki će prostor dobiti oznaku koja mora biti postavljena na vidljivo mjesto i koja će informirati javnost o kvaliteti zraka u tom prostoru. Postoje dvije referentne razine CO_2:

- Razina A propisuje koncentraciju CO_2 u prostoriji manju od 900 ppm ili minimalni ventilacijski tok od 40 m^3 po satu po osobi, uključujući barem 25 m^3 po satu po osobi ventilacije vanjskim zrakom [7].
- Razina B navodi koncentraciju CO_2 u prostoriji manju od 1200 ppm ili minimalni ventilacijski tok s vanjskim zrakom od 25 m^3 po satu po osobi [7].

Što je veći ventilacijski tok, veća je razmjena zraka i stoga manja koncentracija virusa i drugih zagađivača u zraku. Trenutno vlasnici prostora nemaju nikakve obaveze propisane ovim zakonom, ali u budućnosti će biti nužno koristiti mjerač kvalitete zraka, razviti akcijski plan i analizu rizika, prijaviti se za certifikat te ga prikazati na lokaciji.

1.3. Mjerni uređaji

Mjerenje koncentracije CO_2 važno je za osiguravanje adekvatne ventilacije zatvorenih prostora kao i za smanjivanje pretjerane ventilacije te očuvanje energije. Mjerenja koja daju CO_2 senzori mogu koristiti ventilacijski sustavi za automatsko podešavanje volumena

vanjskog zraka potrebnog za održavanje stabilne razine CO_2 i dobre kvalitete zraka. Takva strategija naziva se ventilacija kontrolirana zahtjevom (engl. *Demand controlled ventilation*, skraćeno DCV) i primjenjuje se u prostorima s promjenjivim brojem posjetilaca [5]. CO_2 senzori trebaju se ugraditi na zidove na visini 1 – 2 metra, što dalje od prozora i ventilacijskih cijevi te na udaljenosti oko 2 metra od ljudi [2]. Visoke razine CO_2 pokazuju da prostor nije dobro prozračen što znači da je veći rizik i od ostalih zagađivača zraka te čestica koje prenose viruse ili bakterije. U slučaju povećane koncentracije CO_2 potrebno je povećati ventilaciju, npr. otvaranjem prozora i vrata ili ugrađivanjem zračnih ventila u prostorije, mogu se koristiti i HEPA filtri za uklanjanje aerosola i virusa.

Uređaj za mjerenje kvalitete zraka u zatvorenom prostoru provjerava koncentracije različitih zagađivača u zraku i na temelju toga procjenjuje koliko je zrak siguran za disanje. Upozorava ljude kada koncentracija nekog od zagađivača postane opasna. Mogu imati jedan ili više senzora koji mjere različite zagađivače u zraku te temperaturu i vlagu. Neki uređaji imaju i vizualan prikaz koncentracije određenih zagađivača. S obzirom da trenutno ne postoje široko prihvaćene maksimalne razine zagađivača u zatvorenom prostoru, razine za koje uređaj šalje upozorenje ovise o proizvođaču. Najčešći zagađivači koje uređaji mjere su lebdeće čestice, ugljikov dioksid, ugljikov monoksid, hlapivi organski spojevi, formaldehid i radon [8]. Uređaji za mjerenje kvalitete zraka imaju ograničen životni vijek i tijekom vremena postaju sve manje pouzdani te ih treba redovito mijenjati.

U ovom radu je pretpostavljeno da će se kvaliteta zraka u obrazovnoj ustanovi pratiti pomoću mreže uređaja za mjerenje koncentracije CO_2. Između raznih raspoloživih vrsta senzora, odabrani su nedisperzivni infracrveni CO_2 senzori (NDIR) koji detektiraju prisutnost ugljikovog dioksida na temelju apsorpcije infracrvenog svjetla određenih valnih duljina. Ako je infracrveno svjetlo apsorbirano, CO_2 je prisutan u zraku. Što je više svjetla apsorbirano to je veća koncentracija CO_2. NDIR senzor sastoji se od izvora infracrvene svjetlosti, optičke komore, pojasno propusnog filtra i detektora [10]. Valna duljina filtra koja se koristi za raspoznavanje CO_2 je 4.26 μm [10]. Izvor svjetlosti emitira infracrveno zračenje određene valne duljine koje zatim prolazi kroz optičku komoru. U optičkoj komori nalazi se zrak i ako su prisutne CO_2 molekule, one apsorbiraju određene valne duljine infracrvene svjetlosti, tj. valne duljine koje odgovaraju pojasno propusnom filtru. Detektor mjeri intenzitet infracrvene svjetlosti koja je propuštena iz optičke komore i pretvori ga u električni signal. Taj signal se procesuira kako bi se izolirala apsorpcija uzrokovana prisutnošću CO_2 molekula. Nakon kalibriranja, senzor pruža izlazni signal koji

odgovara koncentraciji CO_2 u zraku. Odnos između transmitancije infracrvene svjetlosti i koncentracije plina opisan je Lambert-Beerovim zakonom:

$$A = \log (I_o/I) = \varepsilon bc \quad (1)$$

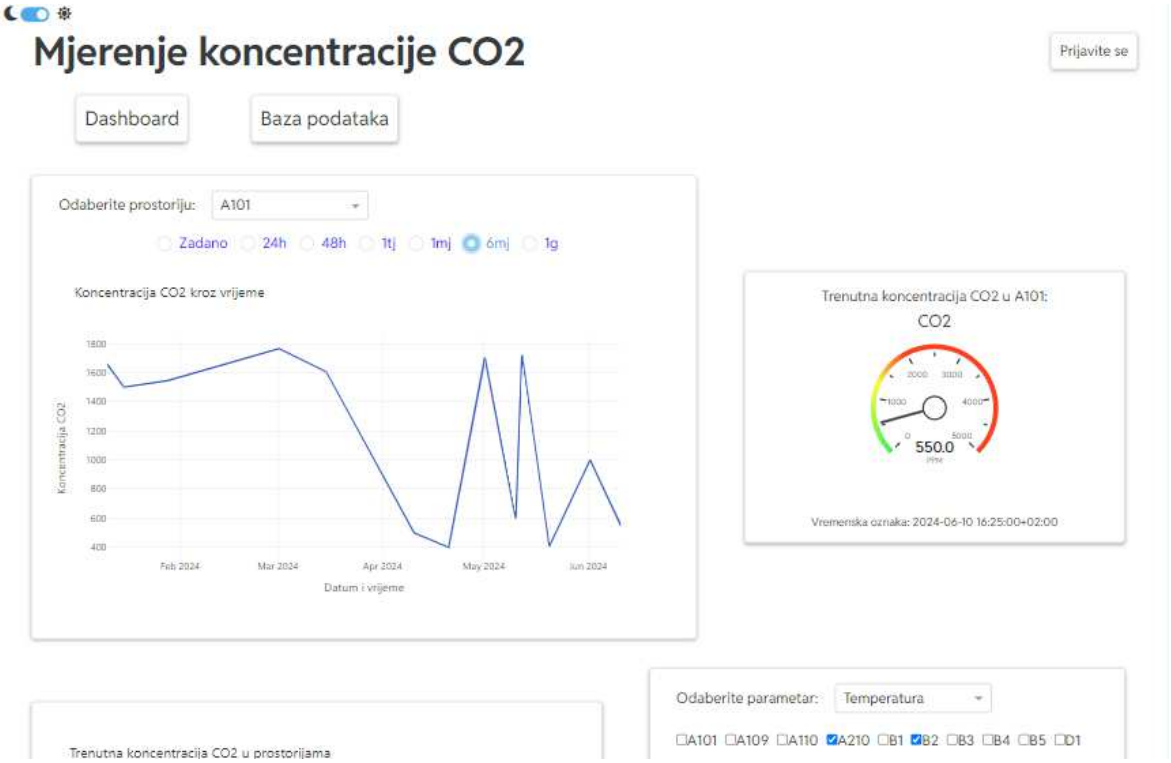
gdje je A apsorbancija na danoj valnoj duljini svjetlosti, ε je molarni apsorpcijski koeficijent, b je duljina puta svjetlosti kroz uzorak, c je koncentracija tvari u otopini, I je intenzitet propuštenog zračenja, a I_o je intenzitet upadnog zračenja [11].

Fotoakustični senzor obrađuje uzorak zraka koristeći pulseve elektromagnetske energije koji su podešeni na apsorpcijsku valnu duljinu od CO_2 [9]. Sa svakim pulsom energije, molekule CO_2 apsorbiraju određenu valnu duljinu i generiraju zvučne valove. Akustični detektor pretvara zvučne valove u očitanja koncentracije CO_2. Elektrokemijski senzori koriste električni naboj za određivanje koncentracije CO_2. Kad CO_2 uđe u senzor, reagira na polimernim površinama i rezultira električnim nabojem [9].

2. Izrada aplikacije

2.1. Funkcionalnost aplikacije

Aplikacija se sastoji od dva dijela: upravljačke ploče koja korisniku nudi interaktivan prikaz podataka dobivenih od mjerača kvalitete zraka i tabličnog prikaza baze podataka s mogućnošću izmjene sadržaja baze. Početna stranica aplikacije prikazana je na slici 2.1.



Slika 2.1 Početna stranica aplikacije

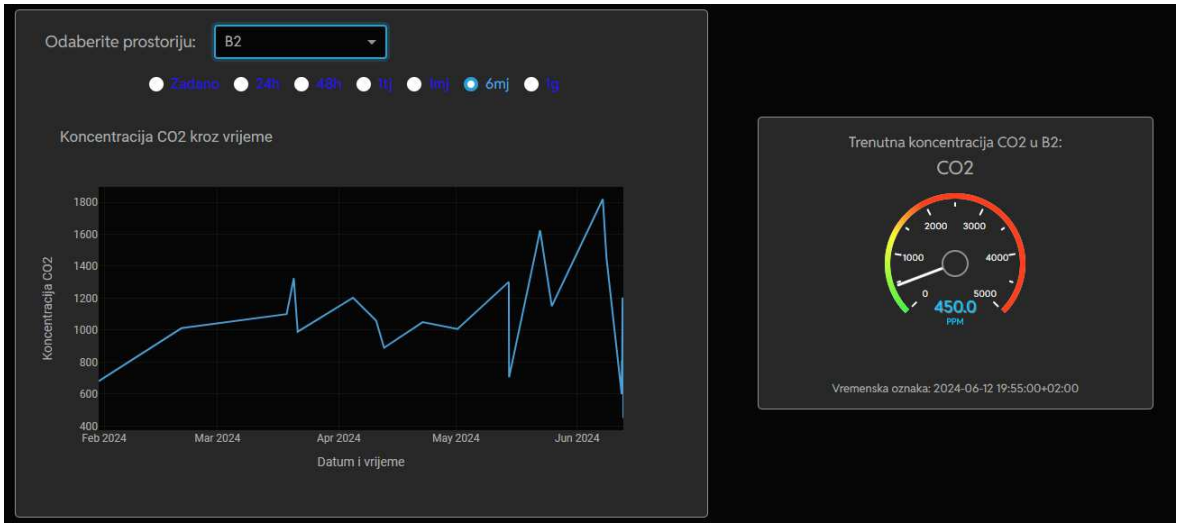
2.1.1. Upravljačka ploča

Na upravljačkoj ploči prikazana su tri grafa i mjerač na kojem je označena trenutna koncentracija CO_2 u zraku u određenoj prostoriji.

Prvi graf prikazuje promjenu koncentracije CO_2 u vremenu za prostoriju koja je odabrana u padajućem izborniku. Također se odabire vremenski period u kojem se želi pratiti promjena koncentracije CO_2 pritiskom na odgovarajući gumb. Korisnik može birati između opcija: zadnja 24 sata, zadnjih 48 sati, zadnjih tjedan dana, zadnjih mjesec dana,

zadnjih šest mjeseci ili godinu dana. Postoji i automatski odabrana opcija zadano koja prikazuje zadnjih 168 mjerenja iz baze podataka.

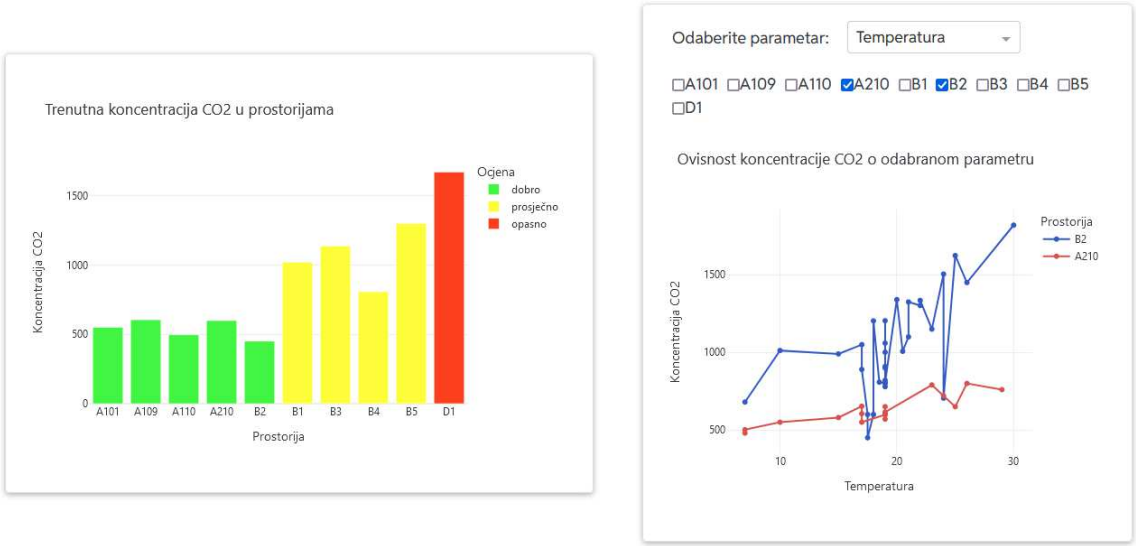
Vrijednost koja se pokazuje na mjeraču također je povezana s prostorijom odabranom u padajućem izborniku. Uzima se zadnja unesena vrijednost iz baze podataka, a vrijeme kada je izmjerena napisano je ispod mjerača. Navedene funkcionalnosti prikazane su na slici 2.2.



Slika 2.2 Graf ovisnosti koncentracije CO_2 o vremenu (lijevo) i mjerač (desno) u prostoriji B2

Drugi graf prikazuje trenutnu koncentraciju CO_2 u svim prostorijama. Pomicanjem kursora po stupcima grafa korisnik može vidjeti točno vrijeme mjerenja i izmjerenu koncentraciju.

Treći graf prikazuje ovisnost koncentracije CO_2 o određenom parametru. Korisnik može odabrati prostorije za koje želi vidjeti graf kao i parametar u padajućem izborniku. Navedeni grafovi mogu se vidjeti na slici 2.3.



Slika 2.3 Graf s trenutnim vrijednostima koncentracije CO_2 u prostorijama (lijevo) i graf ovisnosti koncentracije CO_2 o temperaturi za prostorije B2 i A210 (desno)

2.1.2. Baza podataka

Drugi dio aplikacije čini prikaz tablice iz baze podataka. Kad korisnik pritisne gumb „Baza podataka“ na upravljačkoj ploči, prikaže mu se zadnjih 200 zapisanih mjerenja iz baze po prostoriji. Prije nego što može vidjeti tablicu, korisnik se mora prijaviti, inače dobije odgovarajuću poruku i poveznicu na stranicu za prijavu. Korisnik koristeći interaktivnu tablicu može mijenjati podatke u stvarnoj tablici u bazi podataka. Ima mogućnost brisanja redaka, dodavanja novih redaka i mijenjanja podataka postojećih redaka poput dodavanja komentara za određeno mjerenje. Tablica je prikazana na slici 2.4.

Slika 2.4 Prikaz tablice *co2data* iz baze podataka u aplikaciji

2.2. Korištene tehnologije i alati

Za izradu mrežne aplikacije korišten je radni okvir Dash, a programski kod pisan je u programskom jeziku Python. Kao baza podataka u kojoj su pohranjena mjerenja dobivena od CO_2 senzora koristi se PostgreSQL. Za spajanje s bazom podataka korištene su biblioteke SQLAlchemy i Flask-SQLAlchemy. SQLAlchemy služi za ostvarivanje funkcionalnosti preslikavanja klasa u programskom kodu u odgovarajuće tablice u bazi podataka (engl. *Object Relational Mapping*, skraćeno ORM). Za provjeru identiteta korisnika koji ima pravo unositi podatke u bazu podataka koristi se biblioteka Flask-Login. Za analizu podataka i rad s podatkovnim skupom koristi se biblioteka Pandas. Kako bi se koristile varijable okruženja u kodu, potrebno je učitati modul *os*. Modul *os* sadrži funkcije koje služe za interakciju s operacijskim sustavom. Korištenje varijabli okruženja omogućava pokretanje aplikacije u bilo kojem okruženju bez promjene koda. Također osigurava da podaci za pristup bazi podataka poput korisničkog imena i lozinke nisu hardkodirani [12]. Varijablama okruženja u Pythonu pristupa se preko *os.environ* objekta. U programu se koristi .env datoteka za pohranu potrebnih varijabli okruženja. Kako bi se to moglo, potrebno je koristiti biblioteku python-dotenv.

2.2.1. Radni okvir Dash

Dash je Python biblioteka otvorenog koda za stvaranje interaktivnih mrežnih aplikacija [13]. Dash omogućuje jednostavnu izradu grafičkog korisničkog sučelja za prikaz analitičkih podataka. S obzirom da su Dash aplikacije namijenjene *webu*, sam izgled aplikacije može se lako prilagoditi željama korisnika pomoću CSS-a. Međutim, nije potrebno poznavanje HTML-a ili JavaScripta za izradu aplikacije, već je dovoljno znati Python jer sam Dash nudi niz interaktivnih *web* komponenti. Dash nudi jednostavan reaktivan dekorator za povezivanje korisničkog sučelja i koda za analizu podataka [13]. Taj dekorator zove se *callback* i obično ima argumente *Output* i *Input*. *Input* predstavlja neku komponentu korisničkog sučelja poput padajućeg izbornika (engl. *dropdown*) čija promjena uzrokuje poziv funkcije i time promjenu komponente definirane u *Output* dijelu dekoratora. *Callback* funkcija dobije novu vrijednost ulazne komponente i može izvesti različite operacije poput filtriranja nekog Pandas podatkovnog okvira ili izvođenja SQL upita prema bazi podataka.

Funkcionalnost *web* servera nudi radni okvir Flask, a za izradu frontenda služi React. Flask je jednostavan radni okvir za *web* napisan u Pythonu. React je JavaScript biblioteka za izradu korisničkih sučelja. Komponente u Dashu su Python klase koje odgovaraju određenim React komponentama. Postoje različite biblioteke za Dash komponente kao što su HTML komponente, interaktivne komponente poput grafova i padajućih izbornika, interaktivne tablice i slično. Također je moguće napraviti vlastite komponente. Za vizualizaciju podataka pomoću grafova Dash koristi JavaScript biblioteku Plotly.js.

U izradi aplikacije korištena je i biblioteka dash-bootstrap-components koja nudi svoje komponente te olakšava stiliziranje aplikacije. U Bootstrapu je raspored komponenti definiran preko sustava rešetke [16]. Bootstrap rešetka sastoji se od 12 stupaca što omogućava definiranje različitog ponašanja na ekranima različitih veličina.

Izrada Dash aplikacije

Prvi dio u izradi Dash aplikacije je određivanje rasporeda komponenti (engl. *layout*), koji definira kako aplikacija izgleda korisniku, a drugi je dio pisanje *callback* funkcija koje aplikaciji pružaju interaktivnost. Raspored čini stablo komponenti kojima je definirana struktura aplikacije poput *html.Div* i *dcc.Graph* komponenti. Na slici 2.5. prikazan je kôd za raspored stranice za prijavu, a na slici 2.6. ono što vidi korisnik.

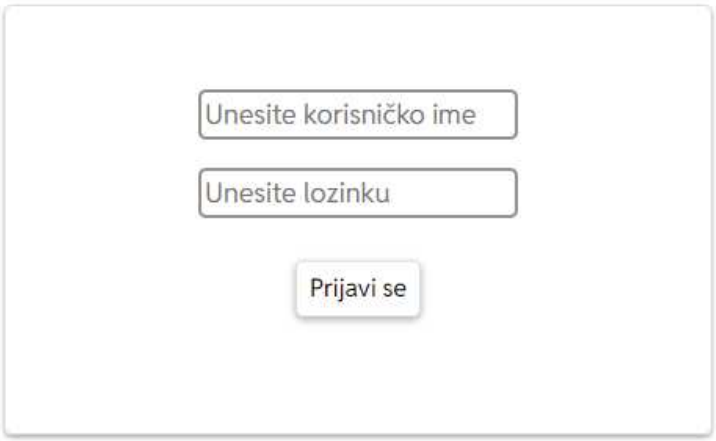
app.py | style.css | show_database.py | db_models.py | home.py | login.py | logout.py

```
import dash
from dash import html, dcc
import dash_bootstrap_components as dbc

dash.register_page(__name__, name="Log In", title="Log In")

layout = dbc.Container(
    [
        dbc.Row( children: [
            dbc.Col( children: [
                dcc.Input(placeholder="Unesite korisničko ime", type="text", id="username-input"),
                dcc.Input(placeholder="Unesite lozinku", type="password", id="passwd-input"),
                html.Button(children="Prijavi se", n_clicks=0, type="submit", id="login-button"),
                html.Div(children="", id="output-state")
            ], className="login-col", xs=12, sm=12, md=12, lg=4, xl=4)
        ], justify='center')
    ]
)
```

Slika 2.5 Kôd za raspored stranice za prijavu prikazan u razvojnom okruženju PyCharm



Slika 2.6 Izgled stranice za prijavu korisnika

Dash.html je modul koji svakoj HTML oznaci ili elementu pridružuje odgovarajuću komponentu u Dash aplikaciji. Na primjer, HTML element *<h1>Hello</h1>*, koji predstavlja naslov najviše razine na *web* stranici, u Dashu možemo generirati koristeći komponentu *html.H1(children='Hello')*. Postoji i modul *dash.dcc* koji služi za interaktivne komponente poput padajućih izbornika, grafova, klizača te lista za odabir. Svaka je komponenta opisana preko pripadnih atributa kao što su *style*, *className* i *id*.

Callback funkcije

Callback funkcije su funkcije koje se automatski pozivaju kad god se promijeni neko svojstvo ulazne komponente, kako bi se promijenila vrijednost nekog svojstva izlazne komponente [14]. Ulazi i izlazi Dash aplikacije definirani su kao argumenti *callback* dekoratora. Dekorator je funkcija u Pythonu koja prima drugu funkciju kao argument i proširuje njeno ponašanje bez mijenjanja te iste funkcije [15]. Dekorator se sastoji od ugniježđene funkcije naziva omotač (engl. *wrapper function*) koja omotava funkciju koju dekorator primi kao argument. Omatajuća funkcija izvodi neki kôd prije i poslije poziva dobivene funkcije i tako mijenja njeno ponašanje.

Ulaz dekoratora u Dashu ima sljedeći oblik: *Input(component_id, component_property)* [14]. *Component_id* odgovara atributu *id* komponente čija promjena uzrokuje promjenu komponente definirane u izlazu. *Component_property* predstavlja svojstvo ili atribut komponente čija promjena uzrokuje aktiviranje *callback* funkcije. *Callback* funkcija kao argument prima novu vrijednost definiranog svojstva ulazne komponente.

Izlaz dekoratora ima sljedeći oblik: *Output(component_id, component_property)*. *Component_id* odgovara identifikatoru komponente čije se svojstvo želi promijeniti pozivom funkcije koju omata *callback* dekorator. *Component_property* predstavlja određeno svojstvo komponente koje se želi promijeniti. Nova vrijednost navedenog svojstva je vrijednost koju vrati *callback* funkcija [14]. Ovo se naziva reaktivno programiranje jer izlazi automatski reagiraju na promjene ulaza.

Callback dekorator smješten je direktno iznad deklaracije same *callback* funkcije u kodu. Također, kao argument dekoratora može se dodati *State*. *State* omogućava da se *callback* funkciji proslijede dodatne vrijednosti atributa nekih komponenti bez njenog aktiviranja.

2.3. Aplikacija

2.3.1. Upravljačka ploča

Kôd za izradu aplikacije podijeljen je u nekoliko modula.

Modul settings.py

Settings.py je modul u kojem se konfigurira Flask server i definira veza prema bazi podataka. Prvo se napravi instanca klase Flask koja predstavlja server i koja se prosljeđuje konstruktoru Dash u modulu app.py. Zatim se postavi tajni ključ za sjednicu i definira trajanje sjednice te kolačića. Stvara se instanca klase SQLAlchemy koja se povezuje s Flask serverom i omogućuje rad s bazom podataka.

Modul co2data.py

U modulu co2data.py definirana je klasa *CO2Data* koja sadrži funkcije za slanje različitih SQL upita bazi podataka. Prvo je potrebno ostvariti vezu s bazom za što služi poziv `create_engine(conn)`. *Engine* je početna točka svake SQLAlchemy aplikacije i osnova za povezivanje s bazom podataka i DBAPI sučeljem [17]. *Enginu* se prosljeđuje URL za bazu podataka koji se nalazi u varijabli okruženja imena *DB_URL*. URL za bazu podataka je string koji definira dijalekt i argumente za povezivanje [17]. Argumenti su korisničko ime, lozinka, poslužitelj i naziv baze podataka, a dijalekt je *postgresql*. Navedeno se može vidjeti u kôdu 2.1. Kreirani *engine* koristit će se pri slanju SQL upita bazi podataka. Funkcija *pandas.read_sql* služi za učitavanje rezultata SQL upita ili cijele tablice iz baze podataka u Pandas podatkovni okvir [18]. Funkcija koja šalje SQL upit bazi

podataka za dohvat najnovijih 168 mjerenja za odabranu prostoriju može se vidjeti u kôdu 2.1.

```
conn = os.environ["DB_URL"]
engine = create_engine(conn)

class CO2Data:
    def make_query_default(self, classroom):
        sql = "SELECT * FROM co2data WHERE
classroom='{classroom}' ORDER BY time DESC LIMIT 168".format(
            classroom=classroom)
        df = pd.read_sql(sql, engine)

        return df
```

Kôd 2.1 Dio sadržaja modula co2data.py

Glavni modul

App.py je modul u kojem se nalazi funkcija *main* koja pokreće Dash aplikaciju na željenom priključku (engl. *port*). Prvo se inicijalizira aplikacija naredbom `app = dash.Dash(…)`, tj. napravi se instanca klase Dash. Zatim se definira raspored aplikacije pomoću komponenti. Bootstrap komponenta *Container* omotava cijeli sadržaj aplikacije. Unutar *Containera* nalaze se redci i stupci, tj. komponente *Row* i *Col*. *Row* komponenta omotava stupce, a stupci omotavaju pojedini sadržaj stranice i određuju koliko će prostora zauzeti na stranici. Argumenti *xs, sm, md, lg, xl* služe za definiranje izgleda stranice na uređajima različite veličine. Na primjer, `xs=12` znači da će sadržaj unutar komponente *Col* zauzeti svih 12 stupaca rešetke na stranici, tj. proširit će se preko cijele dužine ekrana za male mobilne uređaje. Argument `lg=5` označava da će taj isti sadržaj zauzeti pet stupaca rešetke na velikim uređajima poput prijenosnog računala. Argument *justify* kod *Row* komponente služi za određivanje horizontalnog poravnavanja, a argument *align* za određivanje vertikalnog poravnavanja stupaca koji pripadaju nekom retku.

U app.py modulu u prvom retku nalazi se stupac s naslovom upravljačke ploče i stupac s gumbom za prijavu ili odjavu ovisno o statusu korisnika. U drugom retku nalazi se stupac s poveznicama na početnu stranicu, tj. upravljaču ploču i na stranicu s prikazom baze podataka. Aplikacija sadrži i komponentu *ThemeSwitchAIO* [19] za promjenu teme odnosno izbor između svijetlog i tamnog prikaza sadržaja te komponentu *Interval* koja služi za ažuriranje komponenti nakon isteka definiranog intervala. Sadrži i komponentu

Location koja predstavlja adresu u *web* pregledniku te tri *Store* komponente koje služe za pohranu podataka u JSON obliku u *web* pregledniku.

U Dash konstruktoru argument *use_pages* postavljen je na *True* što znači da aplikacija ima više različitih stranica. Sve stranice nalaze se u *pages* direktoriju. U svakom od tih modula nalazi se naredba *dash.register_page(__name__)* koja govori Dashu da je to jedna od stranica u aplikaciji [20]. Sadržaj stranice definira se pomoću varijable ili funkcije naziva *layout*. U rasporedu aplikacije u app.py modulu potrebno je dodati *dash.page_container* na mjesto gdje će se stranice prikazivati. Pri pozivu *dash.register_page(__name__)* definira se put (engl. *path*) koji je povezan s tom stranicom ili će se automatski postaviti prema imenu modula. U modulu home.py postavi se `path='/'` jer se sadržaj stranice *home* želi prikazati kad korisnik ode na */* umjesto */home*. Također se mogu dodati proizvoljni parovi ključ-vrijednost u pozivu pa je tako dodan ključ *link_id* u modulima show_database.py i home.py. Sve stranice koje imaju navedeni poziv nalaze se u registru stranica (engl. *page registry*) za ovu aplikaciju. U modulu app.py iterira se kroz sve stranice u registru i dodaje *Link* komponenta za svaku od njih. Raspored modula app.py prikazan je u kodu 2.2.

```
app.layout = dbc.Container([
    ThemeSwitchAIO(aio_id="theme", themes=[dbc.themes.LITERA,
dbc.themes.CYBORG]),
    dbc.Row([
        dbc.Col(html.H1("Mjerenje koncentracije CO2",
className="header1"), xs=12, sm=12, md=9, lg=10, xl=10),
        dbc.Col(id="user-status-header", className="log-box",
xs=12, sm=12, md=2, lg=1, xl=1)
    ], className="upper", justify='between'),

    dbc.Row([
        dbc.Col(id="link-div", className="page-link",
children=[
            dcc.Link(id=page['link_id'],
children=page['name'], href=page['relative_path'],
className="link") for page in
            dash.page_registry.values()
            if page['path'] not in ('/login', '/logout')
        ], xs=12, sm=12, md=5, lg=5, xl=5)
    ], justify='start'),
    dcc.Interval(id='interval_pg', interval=1200000,
n_intervals=0),
```

```
    dcc.Location(id="url", refresh='callback-nav'),
    dcc.Store(id="url-store"),
    dcc.Store(id="update_store"),
    dcc.Store(id="graph_store"),
    dash.page_container
], className="container-div", id="main-container",
fluid=True)
```

Kôd 2.2 Raspored (engl. *layout*) modula app.py

Početna stranica

U modulu home.py napisan je kôd za stranicu koja služi za prikaz upravljačke ploče s informacijama o koncentraciji CO_2 dobivenim pomoću mreže mjerača.

Raspored stranice

Kao što je već spomenuto, sadržaj svake stranice podijeljen je u 12 zamišljenih stupaca rešetke pa je prema tome oblikovan raspored. Raspored cijele stranice (bez pojedinih atributa zbog veličine koda) vidi se ispod (kôd 2.3). U prvom retku u prvom stupcu nalazi se padajući izbornik za odabir željene prostorije, niz gumba za odabir vremenskog intervala i graf koji vizualizira podatke iz baze podataka na temelju odabira. Graf prikazuje promjenu koncentracije CO_2 kroz odabrani interval u odabranoj prostoriji. U drugom stupcu nalazi se *Gauge* komponenta, tj. mjerač koji prikazuje trenutnu koncentraciju CO_2 u odabranoj prostoriji. Mjerač se nalazi u paketu *dash_daq*. Ovisno o boji na mjeraču možemo vidjeti u kojem rasponu je izmjerena vrijednost. Poviše mjerača nalazi se komponenta *html.Label* koja govori kojoj prostoriji odgovara prikazana vrijednost. Ispod mjerača nalazi se još jedna oznaka (engl. *label*) koja prikazuje točno vrijeme mjerenja.

```
layout = dbc.Container([
    dbc.Row([
        dbc.Col([
            html.Div([
                html.Div([html.Label("Odaberite prostoriju:
", className="label")]),
                html.Div([dcc.Dropdown(id='line_dropdown',
options=list_rooms, value='A101', className="dropdown")],
className="dropdown-div")]),
            dbc.RadioItems(id='radio_items', value='Zadano',
options=['Zadano', '24h', '48h', '1tj', '1mj', '6mj', '1g'],
className="radio-items", inline=True),
```

```
            dcc.Graph(id='my_graph')], xs=12, sm=12, md=12,
lg=7, xl=7, className="graph1-col", id="col1"),
        dbc.Col([
            html.Label(id="gauge-label", children=["Trenutna
koncentracija CO2 u A101: "]),
            daq.Gauge(id="gauge",value=0,label='CO2',
                      showCurrentValue=True, units="ppm",
                      max=5000, min=0),
            html.Label(id="gauge-label2", children=[])
        ], xs=12, sm=12, md=12, lg=4, xl=4, className="gauge-
col", id="col2")
    ], justify='around', align='center'),
    dbc.Row([
        dbc.Col([dcc.Graph(id='my_graph2')], xs=12, sm=12,
md=12, lg=6, xl=6, className="graph2-col", id="col3"),
        dbc.Col([
            html.Div([
                html.Div([html.Label("Odaberite parametar: ",
className="label")], className="label-div"),
             html.Div([dcc.Dropdown(id='line_dropdown_param',
                  options=['Temperatura', 'Broj studenata'],
                  value='Temperatura',className="dropdown")],
                  className="dropdown-div")
            ]),
            dcc.Checklist(id='checklist', value=['A210',
'B2'], options=list_rooms),
            dcc.Graph(id='my_graph3')
        ], xs=12, sm=12, md=12, lg=5, xl=5,
className="graph3-col", id="col4")
    ], justify='around', align='center')
], fluid=True)
```

Kôd 2.3 – Raspored komponenti za početnu stranicu

U drugom retku u prvom stupcu nalazi se graf koji prikazuje trenutnu vrijednost CO_2 u prostorijama, tj. zadnju unesenu u bazu podataka. U drugom stupcu nalazi se padajući izbornik u kojem se odabire parametar prema kojem će se oblikovati graf. Ponuđeni parametri su temperatura i broj studenata kako bi se vidjela ovisnost koncentracije CO_2 o popunjenosti prostorije njima. Ispod padajućeg izbornika nalazi se lista za odabir (engl. *checklist*) na kojoj se obilježe prostorije koje se žele vidjeti na grafu. Posljednja komponenta u drugom stupcu je graf.

Callback dekoratori

Kako bi se ostvarila željena funkcionalnost pri interakciji korisnika s navedenim komponentama, potrebno je definirati *callback* dekoratore. Prvi od *callbackova* kao ulaz prima komponentu `interval_pg`, a kao izlaz komponentu `graph_store`. *Interval* komponenta sadrži atribut `n_intervals` koji označava koliko je puta interval prošao. Vrijednost tog atributa povećava se za jedan svakih `interval` milisekundi. Komponenta `interval_pg` izgleda ovako:

```
dcc.Interval(id='interval_pg', interval=1200000,
n_intervals=0)
```

U ovom kontekstu to znači da će se svakih 1200000 ms (20 minuta) atribut `n_intervals` povećati za jedan i uzrokovati poziv *callback* funkcije. Poziv funkcije rezultira slanjem upita bazi podataka i pohrani dobivenih podataka u korisnikov preglednik pomoću komponente `graph_store`. Ovime se osigurava da se upravljača ploča svakih 20 minuta ažurira s podacima iz baze. Navedeni *callback* izgleda ovako:

```
@callback(Output('graph_store', 'data'),
          Input('interval_pg', 'n_intervals'))
def update_graph_store(n_intervals):
    db_df = CO2Data().make_query()
    return db_df.to_dict('records')
```

Drugi *callback* ažurira vrijednosti u padajućem izborniku prvog grafa i u listi za odabir trećeg grafa. *Callback* aktiviraju dvije ulazne komponente – *graph_store* i *update_store*. Želimo dohvatiti nove vrijednosti iz baze podataka nakon svakog intervala za što služi komponenta *graph_store* ili kad korisnik doda nešto u bazu preko tablice prikazane na putanji */show_database*, tj. kad se promijeni vrijednost pohranjena u *update_store* komponenti. Podaci iz tablice u bazi podataka spremaju se u podatkovni okvir iz biblioteke Pandas. Zatim se podatkovni okvir filtrira naredbom `new_list_rooms = db_df['classroom'].drop_duplicates().tolist()` kako bi se dobile sve prostorije. *Callback* funkcija vrati listu *new_list_rooms* kao vrijednost izlaznih komponenti.

Za svaki od grafova i za mjerač definiran je po jedan *callback*. Za prvi graf ulazne komponente su `graph_store`, `update_store`, `line_dropdown`, `radio_items` i komponenta za promjenu teme. *Callback* funkcija `display_graph` prvo provjerava koji vremenski interval je odabrao korisnik, tj. koja je vrijednost radio

gumba. Ovisno o tome pozivaju se različite funkcije iz klase *CO2Data* za dohvat podataka iz baze. Dohvaćeni podaci spreme se u podatkovni okvir `db_df`. Zatim se vremenska zona u stupcu *time* podatkovnog okvira pretvori u srednjoeuropsku. Nakon toga napravi se nova referenca na podatkovni okvir naziva `df_fig` koja će se koristi u oblikovanju grafa. Iz ulazne komponente `line_dropdown` uzme se vrijednost `dropdown_val`, koju je odabrao korisnik u padajućem izborniku, i doda se u listu `list_val`. Zatim se iz podatkovnog okvira uzmu samo redci u kojima je vrijednost stupca *classroom* jednaka vrijednosti `dropdown_val` i podatkovni okvir se sortira prema vremenu. Ovisno o vrijednosti sklopke za promjenu teme, `template` se postavlja na jednu od Bootstrap tema – Litera ili Cyborg. Atribut `template` služi za definiranje izgleda grafa. Različiti tipovi grafova preuzeti su iz biblioteke plotly.express. Linijski graf definira se pozivom `px.line()`. Atributi `x` i `y` određuju koje će se vrijednosti prikazivati na x i y osi grafa, `labels` služi za promjenu imena osi, `title` služi za postavljanje naziva grafa, a `hover_data` služi za prikaz vrijednosti u tekstualnom okviru pri prolasku kursora preko grafa. *Callback* funkcija vraća definirani linijski graf i tako ga predaje izlaznoj komponenti `my_graph`. Kôd za opisani *callback* vidi se ispod.

```
@callback(Output('my_graph', 'figure'),
          Input('graph_store', 'data'),
          Input('update_store', 'data'),
          Input('line_dropdown', 'value'),
          Input('radio_items', 'value'),
          Input(ThemeSwitchAIO.ids.switch("theme"), "value"))
def display_graph(graph_data, store_data, dropdown_val,
radio_val, toggle):
    if radio_val == 'Zadano':
        db_df = CO2Data().make_query_default(dropdown_val)
    elif radio_val == '24h':
        db_df = CO2Data().make_query_24h()
    elif radio_val == '48h':
        db_df = CO2Data().make_query_48h()
    elif radio_val == '1tj':
        db_df = CO2Data().make_query_1w()
    elif radio_val == '1mj':
        db_df = CO2Data().make_query_1M()
    elif radio_val == '6mj':
        db_df = CO2Data().make_query_6M()
    else:
```

```
        db_df = CO2Data().make_query_12M()
    db_df['time'] = pd.to_datetime(db_df['time'],
utc=True).dt.tz_convert('Europe/Zagreb')
    df_fig = db_df
    list_val = [dropdown_val]
    df_fig =
df_fig[df_fig['classroom'].isin(list_val)].sort_values('time'
)
    template = "litera" if toggle else "cyborg"
    fig = px.line(data_frame=df_fig, x='time', y='co2value',
labels={'time': 'Datum i vrijeme', 'co2value': 'Koncentracija
CO2', 'comment': 'Komentar'}, hover_data={'time': '|%d.%m.%Y.
%H:%M:%S', 'comment': True}, title='Koncentracija CO2 kroz
vrijeme', template=template)
    return fig
```

Kôd 2.4 – *Callback* za ostvarivanje interaktivnosti prvog grafa

Ulazne komponente za drugi graf su `graph_store`, `update_store` i komponenta za promjenu teme. *Callback* funkcija `display_graph_max` prvo provjerava je li sadržaj komponente `update_store` prazan i ako nije, učita se u podatkovni okvir `db_df`. Ako je prazan, to znači da je *callback* funkciju aktivirala promjena komponente `graph_store` i uzima se podatkovni okvir pohranjen u njoj. Zatim se redci podatkovnog okvira grupiraju po prostoriji i iz svake grupe se uzme redak s najnovijom vremenskom oznakom. U podatkovni okvir dodaje se novi stupac naziva `co2color`. On služi za pridavanje ocjene izmjerenoj vrijednosti CO_2. Ako je vrijednost između 0 i 800 ppm, dodjeljuje se ocjena „dobro“. Ako je vrijednost veća od 800 i manja od 1500 ppm, dobiva ocjenu „prosječno“, a ako je veća od 1500 ppm, dobiva ocjenu „opasno“. Zatim se oblikuje stupčasti graf pozivom `px.bar()`. Pomoću atributa `color` određuje se po vrijednostima kojeg stupca iz podatkovnog okvira će se obojati stupci u grafu. *Callback* funkcija predaje definirani stupčasti graf izlaznoj komponenti `my_graph2`. Navedeni *callback* nalazi se u kôdu 2.5. *Callbackovi* za ostale komponente su slični pa se neće dodatno objašnjavati.

```
@callback(Output('my_graph2', 'figure'),
          Input('graph_store', 'data'),
          Input('update_store', 'data'),
          Input(ThemeSwitchAIO.ids.switch("theme"), "value"))
def display_graph_max(graph_data, store_data, toggle):
```

```
    if store_data is not None:
        db_df = pd.DataFrame(store_data)
    else:
        db_df = pd.DataFrame(graph_data)
    db_df['time'] = pd.to_datetime(db_df['time'],
utc=True).dt.tz_convert('Europe/Zagreb')
    df_fig = db_df
    idx =
df_fig.groupby('classroom')['time'].transform('max') ==
df_fig['time']
    df_fig = df_fig[idx]
    co2color = []
    for x in df_fig['co2value']:
        if 0 < x <= 800:
            co2color.append('good')
        elif 800 < x <= 1500:
            co2color.append('avg')
        elif x > 1500:
            co2color.append('bad')
    new_df = df_fig.assign(co2color=co2color)
    new_df = new_df.sort_values('classroom')
    template = "litera" if toggle else "cyborg"
    fig = px.bar(data_frame=new_df, x='classroom',
y='co2value', color='co2color', labels={'co2value':
'Koncentracija CO2', 'classroom': 'Prostorija', 'co2color':
'Ocjena', 'time': 'Vrijeme'}, title='Trenutna koncentracija
CO2 u prostorijama', hover_data={'time': '|%d.%m.%Y.
%H:%M:%S'}, color_discrete_map={"bad": "#fc401e", "avg":
"#fffe3a", "good": "#42f542"}, template=template)
    return fig
```

Kôd 2.5 – *Callback* za prikazivanje drugog grafa

2.3.2. Baza podataka

Klase

Mrežna aplikacija omogućuje prikaz tablice s mjerenjima CO_2 iz baze podataka i izvođenje radnji nad njom. Kako bi se to ostvarilo, potrebno je napraviti klase koje odgovaraju tablicama iz baze podataka. To se obavlja u modulu db_models.py. Definirana je klasa *ValOfCO2* koja odgovara tablici *co2data* u bazi podataka i klasa *User* koja odgovara

tablici *users*. Te klase nasljeđuju klasu *db.Model* iz biblioteke Flask-SQLAlchemy [23]. Za svaku klasu potrebno je definirati atribute koji odgovaraju stupcima u tablici u bazi podataka. Potrebno je definirati i tipove podataka za stupce te integritetska ograničenja preko *db.Column*. To je moguće vidjeti u kôdu 2.6.

```
class ValOfCO2(db.Model):
    __tablename__ = 'co2data'
    __table_args__ = (
        db.UniqueConstraint('time', 'device'),
    )
    id = db.Column(db.BigInteger, primary_key=True)
    co2value = db.Column(db.Float, nullable=False)
    time = db.Column(db.DateTime, nullable=False)
    device = db.Column(db.String(50), nullable=False)
    classroom = db.Column(db.String(20), nullable=False)
    no_students = db.Column(db.Integer, nullable=False)
    temperature = db.Column(db.Float, nullable=False)
    comment = db.Column(db.String(300))
```

Kôd 2.6 – Definicija klase *ValOfCO2*

Shema tablice *co2data* može se vidjeti na slici 2.7. Primarni ključ tablice *co2data* je stupac *id*. *Id* ima tip podataka *bigint* odnosno veliki cijeli broj. U *co2value* i *temperature* pohranjuju se decimalni brojevi (*real*). Tip podataka za stupce *device* i *classroom* je znakovni niz varijabilne duljine s maksimalnom duljinom od 50 znakova (*varchar*), a za stupac *no_students* cijeli broj (*integer*). U stupac *time* pohranjuju se vremenski podaci s vremenskom zonom (*timetz*). Tip podataka za stupac *comment* je *varchar* s maksimalnom duljinom od 300 znakova. Svi stupci osim komentara imaju postavljeno integritetsko ograničenje NOT NULL. Također je postavljeno UNIQUE ograničenje za parove *time* i *device*.

	id [PK] bigint	co2value real	time timestamp with time zone	device character varying (50)	classroom character varying (20)	no_students integer	temperature real	comment character varying (300)
1	1	1080	2024-05-29 17:00:04+02	Dev7	D1	90	26	
2	2	705	2024-04-04 11:45:30+02	Dev2	A109	25	15	
3	3	653	2024-04-03 14:45:45+02	Dev3	A210	30	17	
4	4	1203	2024-04-04 16:25:30+02	Dev4	B2	105	18	
5	5	600	2024-05-10 11:45:02+02	Dev1	A101	19	25	
6	6	800	2024-05-29 14:02:07+02	Dev3	A210	30	26	
7	7	500	2024-05-20 16:45:05+02	Dev2	A109	6	25	
8	8	900	2024-05-12 12:45:02+02	Dev5	B1	120	25	
9	9	655	2024-04-12 11:00:23+02	Dev2	A109	21	17	
10	10	550	2024-04-12 10:00:05+02	Dev3	A210	20	17	

Slika 2.7 Shema tablice *co2data*

U tablici *users* primarni ključ je stupac *username* tipa podataka *varchar(20)*. Tip podataka stupca *password* je *varchar(200)*, a stupac *authenticated* je tipa *boolean*.

Prikaz i promjene sadržaja baze

U show_database.py modulu napisan je kôd pomoću kojeg se prikazuje i mijenja sadržaj baze podataka. Kako bi korisnik mogao vidjeti tablicu i mijenjati ju, prvo se mora prijaviti. Stoga se neprijavljenom korisniku prikazuje odgovarajuća poruka i poveznica na stranicu za prijavu. Prijavljeni korisnik vidi tablicu koja je ostvarena kao *dash_table.DataTable* komponenta. *DataTable* je interaktivna komponenta koja omogućuje sortiranje i filtriranje podataka po stupcima, uređivanje ćelija, brisanje redaka i slično [24].

Na stranici se također nalazi gumb za dodavanje novog retka, gumb za spremanje promjena u bazu podataka i element za prikaz odgovarajuće poruke korisniku. Prvi *callback* služi za ažuriranje vrijednosti u tablici nakon svakog proteklog intervala i prikazivanje poruke o ažuriranju korisniku. Idući *callback* izvodi dodavanje novog praznog retka u tablicu.

Treći *callback* obavlja svu funkcionalnost vezanu uz promjenu sadržaja baze podataka. Ulazna komponenta je gumb *save_to_postgres*, a izlazne komponente su tablica, komponenta za pohranu i element za prikaz poruke korisniku. U *callback* se također prosljeđuje trenutni sadržaj tablice. *Callback* funkcija se aktivira svaki put kad korisnik pritisne na gumb *save_to_postgres*.

U varijablu *db_df* učitava se sadržaj tablice iz baze podataka pomoću funkcije *CO2Data().make_query()* i usporedi sa sadržajem korisničke tablice u varijabli *df*. Ako te dvije tablice nisu jednake, korisnik je napravio nekakve promjene u tablici.

Brisanje

Prvo se provjerava je li obrisan neki redak tako da se usporede identifikatori redaka tablice iz baze i tablice dostupne korisniku. Ako se utvrdi da neki identifikatori nedostaju, bazi podataka šalje se DELETE naredba s redcima za brisanje. U varijablu `sql` zapiše se odgovarajući SQL upit. Za izvođenje transakcije koriste se metode `engine.begin()` i `connection.execute()`, a potvrđivanjem transakcije baza podataka uspješno je ažurirana [21]. To se izvodi sljedećim kodom:

```
sql = """DELETE FROM {table_name} WHERE id in
{deleted}""".format(table_name=os.environ["TABLE_NAME"],
deleted=deleted)
with engine.begin() as connection:
    connection.execute(text(sql))
```

Dodavanje

Zatim se provjerava je li dodan novi redak u tablicu. Dodavanje novog retka ostvaruje se INSERT naredbom u bazi podataka. Još jedan način za interakciju s bazom podataka preko SQL upita je instanciranje SQL sjednice. SQLAlchemy *session* objekt stvara se pozivom tvorca sjednice (engl. *session maker*) kojemu se proslijedi *engine* [22]. Instanca sjednice potrebna je za ostvarivanje ORM-a. U kodu to izgleda ovako:

```
Session = sessionmaker(bind=engine)
session = Session()
```

Varijabli `new_row` pridruži se novi objekt klase *ValOfCO2* čiji atributi poprimaju vrijednost korisničkog unosa. Pozivom `session.add(new_row)` u sjednicu se doda novi objekt odnosno redak, a pomoću `session.commit()` se promjene spreme u bazu podataka i potvrdi se transakcija. Ako je transakcija potvrđena, sve izmjene koje je transakcija napravila postale su trajne, tj. svi SQL upiti su uspješno provedeni i baza je ažurirana novim podacima. `Session.commit()` se izvodi u *try-except* bloku kako bi se uhvatila neka pogreška i ispisala korisniku. To se izvodi sljedećim kodom:

```
new_row = ValOfCO2(id=df.at[row, 'id'], co2value=df.at[row,
'co2value'], time=df.at[row, 'time'], device=df.at[row,
'device'], classroom=df.at[row, 'classroom'],
no_students=df.at[row, 'no_students'], temperature=df.at[row,
'temperature'], comment=df.at[row, 'comment'])
session.add(new_row)
try:
    session.commit()
except Exception as err:
    output = html.P("Pogreška: {})".format(err),
style={'color': '#fc401e'})
```

Na kraju se provjerava je li vrijednost neke ćelije u tablici promijenjena i definira odgovarajuća UPDATE naredba. U komponentu *update_store* spremi se novi sadržaj baze podataka, komponenta *DataTable* se ažurira s novim vrijednostima, a korisniku se vrati

odgovarajuća poruka. Dio koda za ostvarivanje veze s bazom podataka preuzet je s githuba [25].

2.3.3. Autentifikacija

Za ostvarivanje autentifikacije u aplikaciji koriste se biblioteke Flask-Login i bcrypt. Bcrypt je kriptografski algoritam koji služi za generiranje sažetaka lozinki (engl. *hash*), a u Pythonu se implementira pomoću biblioteke bcrypt. Flask-Login služi za upravljanje korisničkim sjednicama, obavljanje funkcionalnosti prijave i odjave korisnika te pamćenja korisničkih sjednica na određeno vrijeme [26]. Najvažniji dio implementacije Flask-Logina je *LoginManager* klasa. Ona definira kako učitati korisnika pomoću identifikatora, gdje poslati korisnika kad se treba prijaviti i slično. Flask-Login koristi korisničke sjednice za autentifikaciju zbog čega je potrebno postaviti tajni ključ aplikacije. Sjednice omogućavaju pohranjivanje informacija specifičnih za nekog korisnika tijekom niza zahtjeva. Klasa *User* koja reprezentira korisnike mora imati definirano:

- svojstvo *is_authenticated* – je li korisnik uspješno autentificiran
- svojstvo *is_active* – je li korisnik aktivan (nije suspendiran i slično)
- svojstvo *is_anonymous*
- metodu *get_id()* – mora vratiti identifikator korisnika tipa string

Definirana klasa *User* odgovara tablici *users* u bazi podataka kao što je prethodno navedeno. Potrebno je definirati *user_loade*r koji se koristi za dohvaćanje *user* objekta, najčešće iz baze podataka, na temelju korisničkog ID-ja pohranjenog u sjednici. Za prijavu korisnika poziva se funkcija *login_user*. Prijavljenom korisniku pristupa se preko objekta *current_user*. Kad korisnik zatvori preglednik, Flask sjednica se obriše i korisnik je odjavljen. Flask-Login nudi opciju „Zapamti me“ koja omogućava da korisnik ostane prijavljen i nakon zatvaranja preglednika. Potrebno je samo proslijediti argument *remember=True* funkciji za prijavu korisnika. Na korisnikovo računalo će se spremiti kolačić, a njegovo trajanje se može ručno postaviti, u ovom slučaju je postavljeno na pet dana.

Modul login.py definira raspored stranice */login* koji se sastoji od polja za unos korisničkog imena, polja za unos lozinke i gumba za prijavu. Kad korisnik unese podatke za prijavu i pritisne gumb „Prijavi se“, aktivira se odgovarajuća *callback* funkcija. *Callback* funkcija prikazana je u kodu 2.7. Unutar funkcije poziva se

`db.session.get(User, username)` koji vraća instancu *usera* na temelju navedenog identifikatora ili primarnog ključa iz baze podataka. U ovom slučaju identifikator je korisničko ime. Za provjeru ispravnosti unesene lozinke koristi se funkcija `bcrypt.checkpw()` koja uspoređuje vrijednost sažetka lozinke pohranjenog u bazi podataka i sažetka lozinke koju je unio korisnik. Ako se vrijednosti podudaraju, vrijednost atributa `authenticated` se postavlja na *True* i mijenja se zapis u bazi podataka pozivom `db.session.add(user)` i `db.session.commit()`. Zatim se poziva funkcija za prijavu korisnika `login_user`.

```
def login_button_click(n_clicks, username, password):
    if n_clicks > 0:
        user = db.session.get(User, username)
        if user:
            user_pass = password.encode('utf-8')
            stored_pass = user.password.encode('utf-8')
            if bcrypt.checkpw(user_pass, stored_pass):
                user.authenticated = True
                db.session.add(user)
                db.session.commit()
                login_user(user, remember=True)
                return "Login Successful"
        return "Login Unsuccessful"
```

Kôd 2.7 *Callback* funkcija za prijavu korisnika

U modulu logout.py prvo se provjerava je li korisnik autentificiran. Ako je korisnik prijavljen, atribut *authenticated* objekta *user* postavlja se na *False* kako bi se označilo da se korisnik odjavio i dodaje se *usera* u SQL sjednicu. Kad se pozove *db.session.commit()*, transakcija se potvrdi i promjene se spreme u bazu podataka. Zatim se pozove funkcija za odjavu korisnika *logout_user()*. Nakon toga, na stranici se prikaže poruka o uspješnoj odjavi s poveznicom na stranicu za ponovnu prijavu. Dio koda za ostvarivanje funkcionalnosti prijave i odjave korisnika preuzet je s githuba [27], a dio s *web* stranice Real Python [28].

2.4. Ograničenja i poboljšanja

U izradi aplikacije koristila se probna baza podataka s oko 200 zapisa u tablici *co2data*, ali u stvarnosti će broj zapisa biti puno veći jer će mjerači slati nove vrijednosti svakih 10 do

20 minuta. Također će se povećati broj prostorija u kojima se provode mjerenja što će utjecati na izgled grafova i prikazane tablice. Ako broj prostorija bude prevelik, moguće je da će prikaz po 200 zapisa za svaku prostoriju preopteretiti komponentu *DataTable*. Kako bi se to spriječilo, moguće je korisniku ponuditi izbor prikaza tabličnih podataka samo za određenu prostoriju, uređaj ili vremenski period, a može mu se ponuditi i odabir broja zapisa za prikazati. To se može ostvariti pisanjem novih prilagođenih SQL naredbi za bazu podataka. U slučaju velikog broja podataka ili velikog broja poziva *callbackova*, može doći do nepotrebnog troška memorije i vremena. Kako bi se to popravilo, *callback* se može modificirati za izvođenje na klijentskoj strani, u pregledniku. *Callback* dekoratoru kao prvi argument doda se JavaScript funkcija koja obavlja istu funkcionalnost kao originalna *callback* funkcija, samo što se izvodi u pregledniku [29].

Pretpostavljeno je postojanje samo jednog korisnika – administratora, koji može pregledavati i mijenjati sadržaj baze podataka. Aplikacija se može unaprijediti tako da se doda autentifikacija korisnika s različitim razinama pristupa i definiraju akcije koje oni mogu izvoditi.

S obzirom da je zadatak bio fokusiran samo na funkcionalnost ovakve aplikacije, za izradu dizajna aplikacije korišten je već gotovi Bootstrap CSS u kombinaciji s prilagođenim CSS-om koji se nalazi u direktoriju */assets*. Međutim, još ima mjesta za poboljšanje estetike upravljačke ploče i bolju prilagodbu za različite uređaje.

Zaključak

Nadziranje kvalitete zraka u zatvorenim prostorima ne smije se zanemariti jer se zbog loše ventilacije u zraku mogu pronaći različiti zagađivači štetni za ljudsko zdravlje. Jednostavan način za mjerenje kvalitete zraka je uporaba CO_2 senzora. Koncentracija CO_2 u zatvorenim prostorima ukazuje na prisutnost ostalih opasnijih zagađivača, a sam CO_2 utječe na koncentraciju i produktivnost što je posebno bitno u obrazovnim ustanovama. Za prikaz mjerenja dobivenih od više različitih mjerača kvalitete zraka u obrazovnom okruženju, napravljena je interaktivna mrežna aplikacija. Za izradu aplikacije koristio se radni okvir Dash za Python koji je znatno olakšao izradu grafičkog korisničkog sučelja i ostvarivanje interaktivnosti korisnika s aplikacijom. Dash nudi niz komponenti za izradu korisničkog sučelja i *callback* dekoratore koji omogućavaju povezivanje akcija korisnika s odgovarajućim funkcijama. Napravljena je upravljačka ploča koja se sastoji od nekoliko grafova s padajućim izbornicima i listama za odabir koji korisniku nude odabir željene funkcionalnosti. Uspješno je ostvareno povezivanje s PostgreSQL bazom podataka korištenjem biblioteka SQLAlchemy i Flask-SQLAlchemy. Zahvaljujući tome, korisniku se prikazuju odabrani podaci u grafovima i tablica s mjerenjima iz baze podataka, a ostvarena je i funkcionalnost dodavanja novih redaka, brisanja redaka te promjene bilo koje vrijednosti iz baze podataka. Provjera identiteta korisnika koji ima pravo mijenjati sadržaj baze podataka ostvarena je korištenjem biblioteke Flask-Login. Jedino uspješno prijavljeni korisnik ima mogućnost mijenjanja sadržaja baze podataka. U slučaju velikog broja zapisa u bazi podataka, aplikacija bi se trebala unaprijediti dajući korisniku više opcija za odabir željenih podataka. Također, aplikacija se može unaprijediti poboljšanjem estetike.

Literatura

[1] EU-OSHA. *Indoor air quality (IAQ)*. Poveznica: https://oshwiki.osha.europa.eu/en/themes/indoor-air-quality-iaq

[2] Peterborough Public Health. *Carbon Dioxide (CO_2) Monitoring for Indoor Air Quality (IAQ)*. Poveznica: https://www.peterboroughpublichealth.ca/wp-content/uploads/2022/03/CO2-Monitoring-for-Indoor-Air-Quality-March-25-2022.pdf

[3] Matić, R. *Izvješće o praćenju emisija hlapivih organskih spojeva u zrak od 2015. do 2017. godine*. Poveznica: https://www.haop.hr/sites/default/files/uploads/dokumenti/011_zrak/Izvjesca/Izvjesce_EHOS_2015.%20do%202017.%20godine.pdf

[4] EPA. *Particulate Matter (PM) Basics*. Poveznica: https://www.epa.gov/pm-pollution/particulate-matter-pm-basics

[5] Prill, R. *Why Measure Carbon Dioxide Inside Buildings?*. Poveznica: https://www.energy.wsu.edu/documents/CO2inbuildings.pdf

[6] Health Canada. *Consultation: Proposed Residential Indoor Air Quality Guidelines for Carbon Dioxide*. Poveznica: https://www.canada.ca/en/health-canada/programs/consultation-residential-indoor-air-quality-guidelines-carbon-dioxide/document.html#appc

[7] Health, Food Chain Safety and Environment. *Legal framework regarding indoor air quality*. Poveznica: https://www.health.belgium.be/en/closer-legal-framework-indoor-air-quality

[8] EPA. *Low–Cost Air Pollution Monitors and Indoor Air Quality*. Poveznica: https://www.epa.gov/indoor-air-quality-iaq/low-cost-air-pollution-monitors-and-indoor-air-quality

[9] Disruptive Technologies. *What Is A CO_2 Sensor And How Does It Work?*. Poveznica: https://www.disruptive-technologies.com/explore/what-is-a-CO2-sensor-and-how-does-it-work

[10] Winsen. *What is a CO_2 Sensor?*. Poveznica: https://www.winsen-sensor.com/knowledge/what-is-CO2-sensor.html

[11] Bajt, P. *Lambert - Beerov zakon*. Završni rad. Sveučilište Josipa Jurja Strossmayera u Osijeku, Odjel za kemiju, 2018. Poveznica: https://repozitorij.kemija.unios.hr/islandora/object/kemos%3A260/datastream/PDF/view

[12] Doppler. *Using Environment Variables in Python for App Configuration and Secrets.* Poveznica: https://www.doppler.com/blog/environment-variables-in-python

[13] Plotly. *Introducing Dash*. Poveznica: https://medium.com/plotly/introducing-dash-5ecf7191b503

[14] Dash Plotly. *Basic Dash Callbacks*. Poveznica: https://dash.plotly.com/basic-callbacks

[15] FreeCodeCamp. *The Python Decorator Handbook*. Poveznica: https://www.freecodecamp.org/news/the-python-decorator-handbook/

[16] Dash Bootstrap Components. Poveznica: https://dash-bootstrap-components.opensource.faculty.ai/docs/components/layout/

[17] SQLAlchemy. Poveznica: https://docs.sqlalchemy.org/en/20/core/engines.html

[18] Pandas. Poveznica: https://pandas.pydata.org/docs/reference/api/pandas.read_sql.html

[19] Plotly. Poveznica: https://community.plotly.com/t/dash-bootstrap-templates-v1-0-0-new-theme-switch-components/58500

[20] Dash Plotly. *Multi-Page Apps and URL Support*. Poveznica: https://dash.plotly.com/urls

[21] SQLAlchemy. Poveznica: https://docs.sqlalchemy.org/en/20/core/connections.html#sqlalchemy.engine.Connection.execute

[22] SQLAlchemy. Poveznica: https://docs.sqlalchemy.org/en/20/orm/session_basics.html

[23] Flask-SQLAlchemy. Poveznica: https://flask-sqlalchemy.palletsprojects.com/en/2.x/models/

[24] Dash Plotly. *Dash DataTable*. Poveznica: https://dash.plotly.com/datatable

[25] Github. Poveznica: https://github.com/alan-navarro/street_team/tree/main

[26] Flask-Login. Poveznica: https://flask-login.readthedocs.io/en/latest/

[27] Github. Poveznica: https://github.com/AnnMarieW/dash-flask-login/tree/main

[28] Real Python. *Using Flask-Login for User Management with Flask.* Poveznica: https://realpython.com/using-flask-login-for-user-management-with-flask/

[29] Dash Plotly. *Clientside Callbacks*. Poveznica: https://dash.plotly.com/clientside-callbacks

[30] Kalinić, N. *Formaldehid u okolišu i njegov utjecaj na zdravlje.* Arhiv za higijenu rada i toksikologiju. Poveznica: https://hrcak.srce.hr/145405

Sažetak

Upravljačka ploča za mrežu mjerača kvalitete zraka

Mjerenje kvalitete zraka u zatvorenim prostorima različito je od mjerenja kvalitete zraka na otvorenom jer uključuje zagađivače uobičajene na otvorenom prostoru, hlapljive organske spojeve, ali i aerosole te plinove povezane s ljudima. Koncentracija ugljikovog dioksida u zatvorenom prostoru izravno utječe na koncentraciju i produktivnost, ali je i dobar pokazatelj sveukupne kvalitete zraka. Cilj ovog rada bila je izrada mrežne aplikacije za interaktivan prikaz mjerenja dobivenih iz mreže mjerača kvalitete zraka u obliku upravljačke ploče. Također je trebalo omogućiti prikaz najnovijih podataka iz baze podataka za prijavljenog korisnika s mogućnošću njihove izmjene. U izradi aplikacije korišten je radni okvir Dash za programski jezik Python. Za povezivanje s bazom podataka korištena je biblioteka SQLAlchemy, a za ostvarivanje prijave i odjave korisnika korištena je biblioteka Flask-Login.

Ključne riječi: mrežna aplikacija, kvaliteta zraka, ugljikov dioksid, Dash, Python, upravljačka ploča, SQLAlchemy, Flask-Login

Summary

Dashboard for air quality sensor network

Measuring air quality in indoor spaces is different than outdoors because in addition to air pollutants typical for the outdoors, like volatile organic compounds, it also contains aerosols and gases associated with people. Indoor CO_2 concentrations have an effect on concentration and productivity but are also a good indicator of overall indoor air quality. The aim of this paper was to create a web application for an interactive display of measurements from a network of air quality sensors in the form of a dashboard. It was also required to enable logged in users to see the display of the newest data from the database with the option of modifying it. The app was built using Dash, a framework for the Python programming language. SQLAlchemy library was used to connect to the database and Flask-Login library was used in order to log in and log out users.

Keywords: web application, air quality, carbon dioxide, Dash, Python, dashboard, SQLAlchemy, Flask-Login