Nadogradnja FER-ove IoT platforme geoprostornim podacima

Zagorc, Marko

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1217

NADOGRADNJA FER-OVE IOT PLATFORME GEOPROSTORNIM PODACIMA

Marko Zagorc

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1217

NADOGRADNJA FER-OVE IOT PLATFORME GEOPROSTORNIM PODACIMA

Marko Zagorc

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 10. ožujka 2023.

ZAVRŠNI ZADATAK br. 1217

Pristupnik:	**Marko Zagorc (0036530572)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Mario Kušek

Zadatak: **Nadogradnja FER-ove IoT platforme geoprostornim podacima**

Opis zadatka:

U današnjem Internetu značajno raste broj jednostavnih uređaja poput senzorskih i aktuatorskih čvorova koji prikupljaju i odašilju podatke s različitih senzora te primaju naredbe koje izvršavaju u fizičkom svijetu. Takvi čvorovi su obično vezani za pojedine stvari i tvore Internet stvari (Internet of Things). U laboratoriju za Internet stvari napravljena je platforma koja služi za povezivanje uređaja u Internetu stvari, spremanje njihovih podatka i kroz aplikacijsko programsko sučelje dohvaćanje podataka pomoću REST sučelja. Vaš je zadatak nadograditi postojeće podatke o IoT uređajima u FER-ovoj IoT platformi s podacima o lokaciji. Kao repozitorij za podatke koristite bazu podataka PostgreSQL s dodatkom PostGIS. Potrebno je napraviti klijentsku web-aplikaciju koja će prikazivati kartu s IoT uređajima i na odabir uređaja pokazati konkretne podatke za vremenski period. Web-aplikacija komunicira s poslužiteljem pomoću REST sučelja s podrškom za geoprostorne objekte u formatu GeoJSON. Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Rok za predaju rada: 9. lipnja 2023.

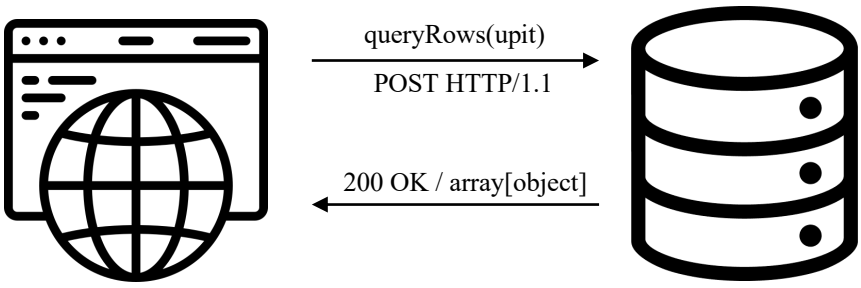
SADRŽAJ

UVOD

Bitnu stavku u današnje doba, bavimo li se bilo kakvom gospodarskim sektorom, predstavljaju trenutačno dostupni, relevantni i precizno oblikovani podatci koji su spremni za manipulaciju. FER-ova IoT platforma sa svojim entitetima, odnosno umreženim senzorima zahtjeva ažurnost, kao i informaciju gdje entitet obavlja mjerenja. Shodno tome, problematiku ovog rada se može podijeliti na dva problema – prikaz geografske pozicije entiteta te dostupnost podataka njegovih mjerenja s ispravnom reprezentacijom istih. Kako bi se riješilo pitanje dostupnosti podataka, prilikom korištenja rezultata ovog Završnog rada – klijentske aplikacije, moguće je pregledati te filtrirati sve senzore koji su u sustavu FER-ove IoT platforme po vlastitom odabiru, a zatim očitati njihova mjerenja te ih grafički prikazati kako bi se riješio geopozicijski dio problematike samog rada. Kompletna aplikacija napisana je u programskom jeziku Javascript u okruženju Node.js koje osigurava ispravno izvođenje aplikacije. Sama aplikacija je dizajnirana u dva dijela, a to su korisničko sučelje koje vrši prikaz web stranice u pregledniku dok drugi dio predstavlja poslužitelja koji ostvaruje vezu s bazom podataka, odgovara na zahtjeve korisnika te dostavlja tražene podatke. Aplikacija ima svojstvo skalabilnosti i korištenje je vrlo intuitivno. Tijekom daljnje razrade Završnog rada, svaka komponenta aplikacije je zasebno odvojena u poglavlja i prateća potpoglavlja kako bi se postiglo što bolje razumijevanje strukture same web-aplikacije.

1. Baza podataka InfluxDB

Kako bi se mjerni uređaji povezali u smislenu cjelinu, njihova očitanja se moraju skladištiti na za to prikladno mjesto. Za IoT platformu preporučljivo je korisititi se bazom podataka sa stvarnovremenskim značajkama, a tijekom izrade aplikacije je korištena inačica takve baze podataka InfluxDB[1]. Stvarnovremenske značajke su svojstva baze podataka da nakon kratkih vremenskih perioda očita traženu mjernu vrijednost sa senzora i adekvatno ju spremi. Baza je implementirana da jednakim performansama i dostavlja mjerenja koje zahtjeva korisnik iste. Dopušta HTTP POST zahtjeve s upitom na bazu u tijelu samog zahtjeva pomoću integriranih funkcija unutar biblioteke koje su programirane da ostvare komunikaciju s bazom Influx. Takve su biblioteke također dostupne za više programskih jezika poput C#-a, Javascripta, PHP-a, Pythona i sl. Tijekom izrade aplikacije korištena je biblioteka „@influxdata/influxdb-client“ koja stvara vezu s bazom, inicijalizira novog „klijenta“ na bazi, a zatim uz pomoć funkcije „queryRows“ s odgovarajućim parametrima šalje upit na istu. Baza odgovara objektom koji predstavlja jedno očitanje senzora, odnosno odgovara poljem objekata ukoliko postoji više očitanja za traženi upit. Komunikacija s bazom podataka ilustrirana je na slici (Sl. 1.1) koja se nalazi u nastavku.



Slika 1.1 Komunikacija web-aplikacije s bazom podataka InfluxDB

Programsko ostvarenje komunikacije započinje uključivanjem potrebne biblioteke u datoteku, u okviru ove aplikacije to je „@influxdata/influxdb-client“, potom inicijalizacija objekta klijenta, nakon toga slijedi pisanje upita u <String> formatu te završava pozivom funkcije „queryRows“ koja kao parametar prima navedeni upit.

1 https://www.influxdata.com/, web stranica baze podataka InfluxDB

Implementacija u programskom jeziku Javascript prikazana je na slici (Sl. 1.2).

```
const {InfluxDB, flux} = require('@influxdata/influxdb-client')

        const client = new InfluxDB({url: URL, token: TOKEN})
        const queryApi = client.getQueryApi(ORGANISATION)

        const query = flux`from(bucket: "${bucket}")
          |> range(start: ${time} )
          |> filter(fn: (r) => ._measurement == "${measurement}")`

        queryApi.queryRows(query, {
            next(row, tableMeta) {
                const o = tableMeta.toObject(row)
                console.log(`${o._time} ${o._measurement}: ${o._field}=${o._value}`)
            },
            error(error) {
                console.error(error)
                console.log('Finished ERROR')
            },
            complete() {
                console.log('Finished SUCCESS')
            },
        })
```

Slika 1.2 Implementacija InfluxDB klijenta u Javascript-u

Bitne stavke u programskoj implementaciji su URL na kojem se nalazi API za odgovaranje na HTTP POST zahtjev, TOKEN za autorizaciju, BUCKET kao instanca baze podataka iz koje povlačimo podatke te ORGANISATION kao naziv organizacije koja polaže prava nad tom instancom baze podataka. Upitni jezik koji se koristi je Flux, a kao najvažniji segmenti upita navedeni su:

- from() – definira BUCKET,
- range() – definira vremenski raspon,
- filter() – definira uvjet upita, poput WHERE u SQL,
- group() – definira grupiranje po zadanom ključu.

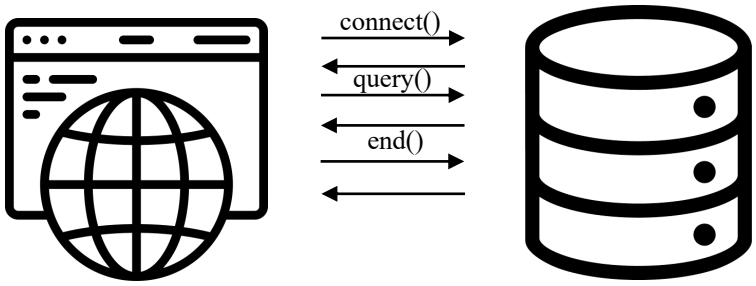
Ova baza podataka podržava i druge upitne jezika, ali s prethodnom instalacijom potrebnih *driver*-a.

2. Baza podataka PostgreSQL s dodatkom PostGIS

Kao repozitorij za podatke korištena je objektno-relacijska baza podataka PostgreSQL sa ekstenzijom PostGIS kako bi se ostvarila mogućnost filtriranja krajnjih uređaja po karti u ovisnosti s lokalitetom koji odaberemo. U sklopu aplikacije implementirano je filtriranje po županijama radi bržeg i efikasnijeg prikaza.

2.1 Baza podataka PostgreSQL

Baza podataka PostgreSQL[2] je strukturirana kao objektno orijentirana te objektno-relacijska što daje dodatnu jednostavnost kreiranju aplikacija koje zahtjevaju obradu nad većim skupovima podataka, a koji su objektne prirode. Svaka izmjena u bazi naziva se transakcijom koja zadovoljava ACID standarde, odnosno standarde poput atomarnosti, konzistentnosti, izolacije te izdržljivosti. PostgreSQL omogućuje korisniku kreaciju različitih baza, shema, tablica te ograničenja nad pojedinim stavkama baze ili pojedine tablice. Uz to pruža i mogućnost instaliranja ekstenzija koje bi omogućile efikasniji zapis podataka i preciznije opisale zadani model. Neke od zanimljivijih su definitivno PostGIS za spremanje geoprostornih podataka te PostPic za spremanje slika. Manipulacija podataka nad bazom u razvoju aplikacija odvija se preko programskih modula, odnosno skupa biblioteka koje inicijaliziraju komunikaciju s bazom te izvršavaju zadane upite. U nastavku će se dodatno pojasniti korištenje modula „pg“ te njegovog segmenta Client, a međusobna komunikacija je ilustrirana na slici (Sl. 2.1).

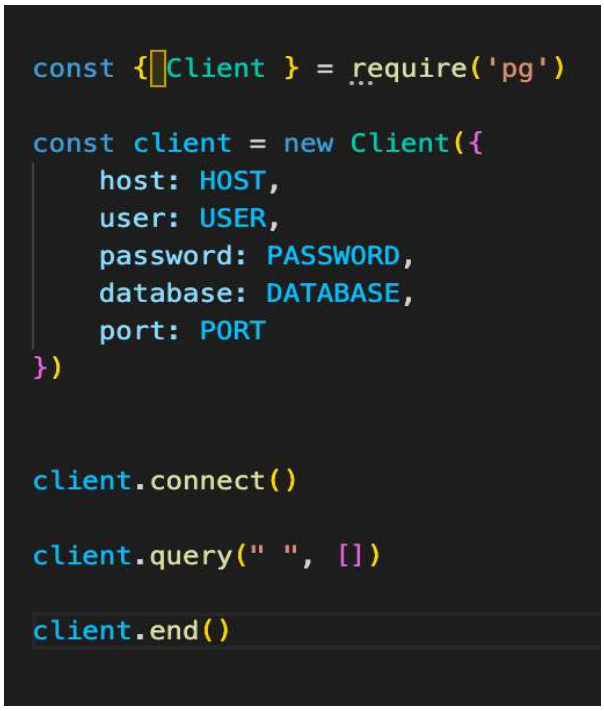


Slika 2.1 Komunikacija web-aplikacije s bazom podataka PostgreSQL

[2] https://www.postgresql.org/, web stranica baze podataka PostgreSQL

2.1.1 Komunikacija s bazom podataka

Komunikacija je ostvarena pomoću modula „pg“ koji stvara aktivnu konekciju (klijenta) na bazu, izvršava niz upita, vraća rezultate upita te završava rad gašenjem servera, odnosno konekcija se prekida i komunikacija završava. Drugim riječima, dokle god je server dostupan, dostupni su i upiti nad bazom podataka, osim ako zbog nekih eksternih okolnosti baza postane nedostupna. U nastavku na slici (Sl. 2.2) je prikazano programsko ostvarenje navedenog.



Slika 2.2 Stvaranje klijenta na bazi PostgreSQL u Javascript-u

Isječak programskog koda prikazuje 3 faze ostvarenja komunikacije, a to su redom: registracija zadanog modula, inicijalizacija klijenta sa zadanim parametrima te izvršavanje upita. Izrazito je bitno ne izostaviti niti jedan dio programskog koda jer može doći do ozbiljnih problema prilikom realizacije komunikacije s bazom te pokretanja servera. Ukoliko je faza registracije i inicijalizacije ispravno napravljena, dolazi faza izvođenja upita uz pomoć funkcije „query()“ koja kao parametre prima upit u <String> formatu te polje vrijednosti ako se koristi dinamički zapis pa želimo predati određene varijable od interesa. Komunikacija se završava metodom „end()“.

2.2 Geografski podatci i dodatak PostGIS

Jedna od najpoznatijih ekstenzija za PostgreSQL bazu podataka je PostGIS koji proširuje mogućnosti navedene baze spremanjem, indeksiranjem i dohvaćanjem podataka s geoprostornim svojstvima.[3] Naime, omogućava manipulaciju prostornih podataka različitim funkcijama, bilo to u 2D ili 3D. Podatci su uz pomoć PostGIS-a prikazani pomoću različitih tipova, npr.:

- Point – točka, (x, y[, z]),
- Line – linija, skup točaka,
- Polygon – poligon, skup linija,
- Multi-geometries – kombinacija svega od navedenog.

Različitim integriranim metodama možemo odrediti odnose između pojedinih objekata, recimo udaljenost dvaju objekata, zatim nalazi li se prostor nekog objekta u prostoru nekog drugog, sijeku li se ta dva prostora ili dodiruju li se. Neke od njih su:

- ST_Intersection() – presjek dvaju prostora,
- ST_Contains() – nalazi li se prostor objekta u prostoru drugog,
- ST_Distance() – udaljenost dvaju objekata u metrima,
- i sl.

Budući da je PostGIS-ovo spremanje podataka organizirano u obliku Kartezijevog sustava, nailazimo na prepreku pri pretvaranju podataka iz svijeta i života u tablični zapis, točnije koordinata po geografskoj širini i dužini. Radi toga preporuča se korištenje formata geoJSON[4] koji predstavlja adekvatan format spremanja geoprostornih podataka u bazu PostgreSQL.

2.2.1 Pohrana potrebnih modela uz geoJSON

GeoJSON je široko upotrjebljavan tip JSON formata podataka koji pruža specifičan opis podataka geografske strukture. Omogućava lak zapis objekata s njihovim

[3] http://postgis.net/, web stranica ekstenzije PostGIS
[4] https://geojson.org/, web stranica formata GeoJSON

prostornim svojstvima unutar tablice u bazi podataka. Tijekom izrade aplikacije svi senzori su pretvoreni u geoJSON format te kao takvi pohranjeni u bazu.

Primjer jednog senzora zapisanog u geoJSON formatu prikazan je na slici (Sl. 2.3) u nastavku.

Slika 2.3 Prikaz senzora u geoJSON formatu

Komponente ovog geoJSON modela su „type“, deklarativni naziv za značajku, „properties“, svojstva objekta – ovdje spremamo ID, naziv senzora i koordinate, a zatim „geometry“ koji omogućava PostGIS-u adekvatno indeksiranje objekta, njegovo spremanje u bazu podataka te na kraju dopušta manipulaciju tog objekta uz pomoć integriranih metoda. Spremanje podataka u geoJSON formatu u bazu podataka PostgreSQL obavlja se preko *Command Line Tool*-a na Windowsu, odnosno *Terminal*-a na Macu. Koristi se naredba „ogr2ogr“ čija je glavna funkcija pretvorba datoteke iz jednog formata u drugi. Puna naredba dana je u nastavku:

- *ogr2ogr -f PostgreSQL PG:“host=HOST user=USER password=PASSWORD dbname=DATABASE“ /fullPath/senzori.geojson“ .*

Velikim slovima napisani su varijabilni parametri ove naredbe. *Host* definira URL na kojem se pristupa bazi podataka, *user* i *password* su definirani podatcima administratora baze podataka, a *dbname* je naziv baze podataka kojoj se pristupa preko zadanog URL-a. Nakon određenih parametara piše se puna putanja do datoteke koju želimo pretvoriti u tablicu unutar baze podataka.

Također, ista naredba se mora primijeniti za spremanje podataka o administrativnim granicama gradova, općina i županija. Tražena datoteka se preuzima s GitHub-a preko sljedećeg URL-a:

- https://github.com/codeforcroatia/open-data/blob/master/adm_granice_gradovi_opcine/OSMB-d092bf8fc94f5319b6a3eb9acaea3b124fd01691.geojson.

3. Arhitektura i dizajn programskog rješenja

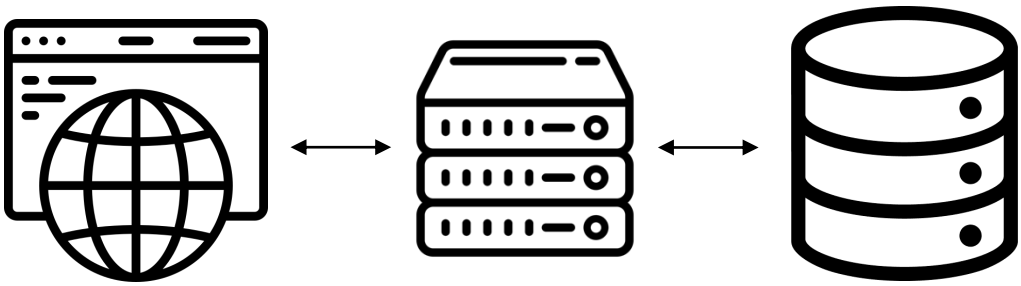
3.1 Specifikacija programske potpore

Prije početka razvoja aplikacije treba definirati točne zahtjeve koje želimo da naša aplikacija obavlja. Prilikom izrade aplikacije za FER-ovu IoT platformu kao dva funkcionalna zahtjeva određeni su prikaz senzora na karti uz i grafička prezentacija mjerenja s pojedinog senzora. Prikaz senzora na karti omogućava jednostavniju vizualizaciju cijele FER IoT mreže koja pruža i opciju filtriranja što također ubrajamo u potrebnu specifikaciju programske potpore. Sljedeći zahtjev, tj. grafički prikaz senzornih očitanja obrađuje se uz pomoć linijskih grafikona za zadane vrijednosti, odnosno relacije vrijednost – vrijeme.

3.2 Arhitektura

Nakon definiranja temeljnih funkcionalnosti aplikacije, idući korak je odabir prikladnog dizajna i modela arhitekture. Najbolji odabir bio je *Client – Server - Database* model aplikacije. Čitav sustav je raspodijeljen u 3 podsustava koji svaki ima svoju jedinstvenu zadaću. Pa tako slijedom, *Client* je zapravo web-aplikacija koja na zahtjev korisnika putem HTTP-a dohvaća resurse sa *Servera*, odnosno web-poslužitelja koji na taj zahtjev odgovara odgovarajućim resursom. U ovoj aplikaciji resursi koji se dohvaćaju

su spremljeni kao zapisi u tablicama unutar baze podataka, a svaki zapis u tablici predstavlja jedan senzor u stvarnom svijetu. Arhitektura je ilustrirana slikom (Sl. 3.1) u nastavku.



Slika 3.1 Ilustracija arhitekture aplikacije

3.3 Implementacija poslužitelja

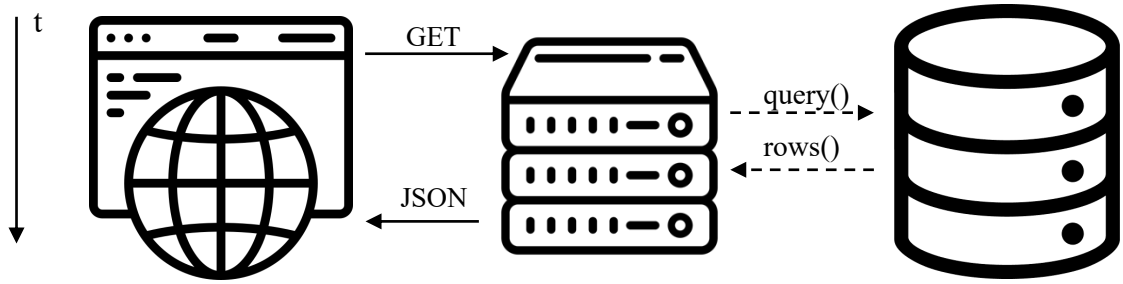
U aplikaciji kao produktu ovog Završnog rada web poslužitelj implementiran je kao dinamički web poslužitelj, odnosno u mogućnosti je generirati nove i dohvatit već pohranjene podatke iz baze podataka.

3.3.1. Razvoj poslužitelja i aplikacijsko programsko sučelje

Backend ove aplikacije, tj. web-poslužitelj konfiguriran je u radnom okviru Express[5] u programskom jeziku Javascript. Shodno tome, unutar datoteke server.js definiran je modul radnog okvira, definiran je i port na kojem server osluškuje zahtjeve od web-klijenta te su definirane funkcije koje se izvršavaju kad na određeni url stigne GET zahtjev. Također, konfiguracija poslužitelja sadržava „*cross-origin resource sharing*“ mehanizam kojim se dopušta jedino da s *frontend* dijela aplikacije dolaze zahtjevi na server. Time je aplikacija obogaćena za dodatnu stopu sigurnosti i integriteta podataka jer niti jedan drugi entitet ne može poslati zahtjev na neki od URL-ova web poslužitelja. Preciznije, unutar aplikacije realizirana su dva jedina moguća odgovara na GET zahtjev ovisno o tome na koji URL stignu. Ukoliko zahtjev stigne na URL *http://backend_host:port/api/:county*, funkcija koja se pritom pozove pročita parametar

[5] https://expressjs.com/, web stranica radnog okvira Express

iz URL-a, izvrši upit nad bazom podataka s tim parametrom - u ovom slučaju parametar predstavlja naziv županije, te vrati pošiljatelju GET zahtjeva tražene rezultate upita - u ovom slučaju to su senzori koji se nalaze na području te županije. Drugi implementirani slučaj je kad GET zahtjev stigne na URL *http://backend_host:port/api/sensors/:id*. Tada se poziva metoda koja čita parametar iz URL-a, ona izvršava upit nad bazom podataka s tim parametrom - u ovom slučaju radi se o ID-ju senzora, potom pošiljatelju GET zahtjeva vraća rezultate upita - u ovom slučaju to je jedan objekt, odnosno jedan senzor s odgovarajućim ID-jem. Sveobuhvatna komunikacija prikazana je na slici (Sl. 3.1) u nastavku.



Slika 3.1 Komunikacija među podsustavima aplikacije

3.4 Korisničko sučelje

Korisničko sučelje je dodirna točka aplikacije i krajnjeg korisnika. Predstavlja vidljivi dio aplikacije, odnosno skriva poslovnu logiku aplikacije, programsku implementaciju komunikacije s poslužiteljem te omogućuje korisniku lakšu i jednostavniju interakciju. Naime, korisničko sučelje možemo proširiti tako da dodamo grafičku aspekt, pa se dio koji dolazi u dodir s korisnikom naziva i grafičkim korisničkim sučeljem.

3.4.1 Posluživanje web-stranice u okruženju React

Za razvoj grafičkog korisničkog sučelja odabran je radni okvir React[6] zbog svojih širokih mogućnosti koji programiranje čine lakšim i razumljivijim te pruža efikasnije programiranje jer dugačke JavaScript kodove može zamijeniti gotovim integriranim metodama. React se svodi na kreiranje komponenata kao osnovnih gradivnih jedinki koje se spajaju u jednu logičku cjelinu. Pojedina komponenta može sadržavati druge komponente, odnosno dubina ugniježđenja je neograničena. Svaka komponenta realizirana je u obliku funkcije koja za povratnu vrijednost ima HTML kod što pridodaje na čitljivom pisanju HTML-a popraćenog React metodama koje interakciju na stranici čine lakšom za programiranje. Definirano je ukupno pet komponenata u sklopu ove aplikacije, a to su redom: *Home*, *Header*, *Map*, *Device* i *Measurement*. Intuitivnog su naziva te imenovani engleskim inačicama. *Header* obnaša estetičku funkcionalnost prepoznatljivosti aplikacije s logom FER-a te sadrži putanju za povratak na kartu. Prisutan je tijekom prikaza svih ostalih komponenata. *Map*, kako i ime sugerira, prikazuje kartu Hrvatske te osigurava prikaz senzora. *Home* sadrži komponentu *Map* i opciju filtriranja implementiranu kao „*search-bar*“ koja omogućuje upis županije i zadovoljava jednu od specifikacija programske potpore. Nakon filtriranja po karti te odabira pojedinog senzora, na red dolazi komponenta *Device*. Navedena prikazuje opće informacije o pojedinom senzoru, npr. naziv, ID te koordinate. Sadrži gumb koji kao rezultat klika prikazuje komponentu *Measurement*. Ova komponenta je realizacija reprezentativnog dijela specifikacije programske potpore, odnosno omogućava korisniku pregled mjerenja za odabrani senzor.

3.4.2 Usmjeravanje na web-stranici

Jedan od glavnih segmenata razvoja web-stranice, odnosno grafičkog korisničkog sučelja je ispravno definiranje ruta, odnosno usmjeravanja tijekom korištenja web-aplikacije. Složen je proces ukomponirati pojedine komponente s jedinstvenim URL-ovima kako bi se postigla smislena cjelina. U aplikaciji su definirane iduće rute:

[6] https://react.dev/, web stranica radnog okvira React

- *http://frontend_host:port/* – na ovoj ruti postavljena komponenta Home,
- *http://frontend_host:port/devices/:id* – na ovoj ruti postavljena komponenta *Device,*
- *http://frontend_host:port/devices/:id/measurements* - na ovoj ruti postavljena komponenta *Measurement*.

Za realizaciju usmjeravanja iskorišten je Javascript paket „*react-router-dom*“ koji sa svojim komponentama *BrowserRouter* – upravlja usmjeravanjem, *Routes* – deklarira sve rute, te *Route* – instanca pojedine rute, omogućava ispravan rad web-aplikacije na različitim URL-ovima.

3.4.3 Vanjske biblioteke korištene za razvoj korisničkog sučelja

Kako bi se mogli zadovoljiti početni zahtjevi aplikacije, bilo je bitno odabrati ispravan način za realizaciju istih. Iz tih razloga, odabrane su vanjske biblioteke Leaflet.js[7] te Chart.js.[8] Prva od navedenih, tj. Leaflet.js iskorištena je za izradu karte i prikaz senzora na karti. Laka je za korištenje i pruža skalabilno rješenje za velike skupove podataka. Moguća je slojevita podjela karte što pogoduje lakšoj reprezentaciji podataka te prikazu različitih ulaznih podataka na istoj karti. Iduća vanjska biblioteka korištena pri izradi aplikacije jest Chart.js koja nudi široki spektar grafova koji se mogu programski implementirati, no u ovoj aplikaciji odabran je upravo linijski graf zbog prirode podataka koji će se na njemu prikazivati. Nadalje, izbor Chart.js-a bio je utemeljen na intuitivnom korištenju i jednostavno razumljivim parametrima koji su neophodni za prikaz traženoga grafa. Chart.js je javno dostupna biblioteka otvorenog koda.

4 Korištenje aplikacije i analiza rješenja

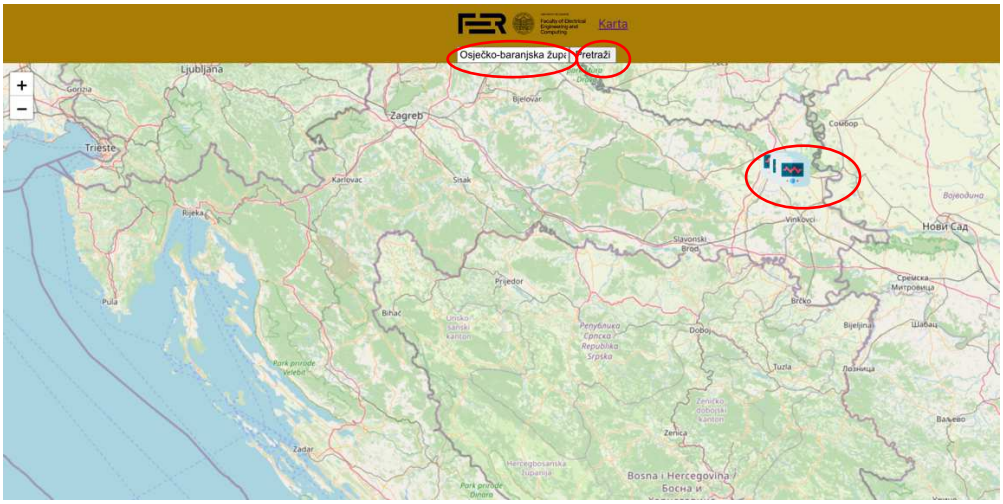
Prilikom razvoja aplikacije korišten je Visual Studio Code kao razvojno okruženje s ekstenzijama za bolju preglednost i čitljivost koda te ekstenzija *Live Server* koja pruža

[7] https://leafletjs.com/, web stranica biblioteke Leaflet.js
[8] https://www.chartjs.org/, web stranica biblioteke Chart.js

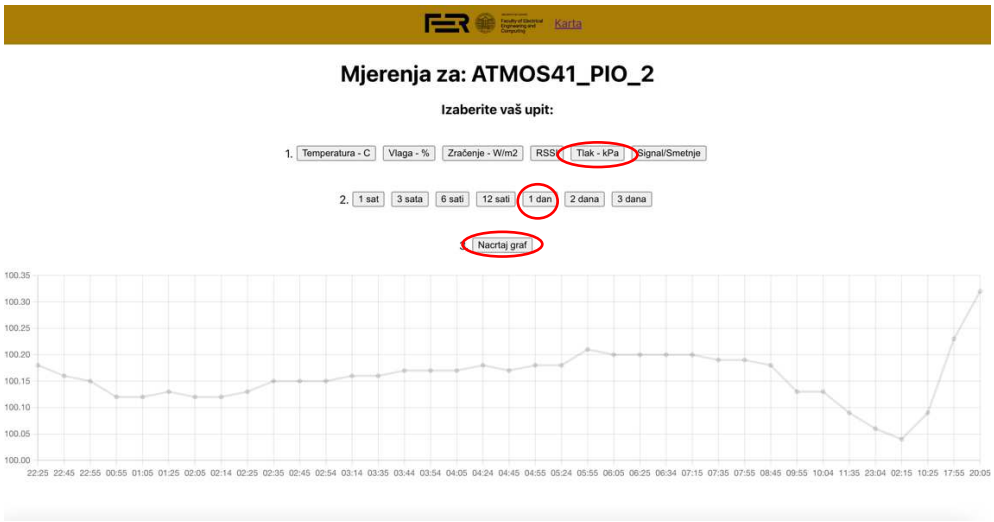
trenutačno osvježenje web-stranice nakon promjene u kodu kako bi se komponente mogle efikasno konstruirati, a shodno tome i pratiti promjene u dizajnu. Prilikom testiranja korisničkog sučelja, uočeno je da prilikom upita na bazu InfluxDB za odabrani senzor ne postoje mjerne vrijednosti u zadanom vremenskom rasponu ili za pojedini predmet mjerenja (temperatura, vlaga itd.). Taj dio se označio kao greška o kojoj treba pravovremeno obavijestiti korisnika. Nadalje, poslužiteljski dio aplikacije testiran je uz pomoć vanjskog servisa Postman koji šalje upite na zadani URL i prikazuje HTTP odgovor. Prilikom takvog testiranja, ustanovljeno je da za neke tražene parametre ne postoje zapisi u bazi podataka, odnosno ne postoje senzori koji odgovaraju traženom parametru pa se taj dio označio kao greška o kojoj treba obavijestiti korisnika. Dvama mogućim greškama se adekvatno rukuje, tj. osmišljen je intuitivan način da se korisnika obavijesti da za traženi upit nema rezultata, pa tako prilikom filtriranja senzora po županijama pojavljuje se ispis „Nema senzora u traženoj županiji“, a u slučaju da za odabrani senzor ne postoje mjerenja korisniku se prikazuje poruka „Nema mjerenja za traženi upit.“ Dodatnim testiranjem utvrđeno je da aplikacija zadovoljava početne specifikacije programske potpore i da ispravno odgovara na zahtjeve korisnika. Ispravno korištenje aplikacije bi izgledalo ovako:

1. korisnik upisuje županiju za koju želi pregledati postojeće senzore
2. a) ne postoje senzori u traženoj županiji, aplikacija ispisuje odgovarajuću poruku
   b) postoje senzori u traženoj županiji te ih aplikacije prikazuje na karti
3. korisnik odabire željeni senzor, a aplikacija ga usmjerava na stranicu odabranog
4. korisnik pregledava informacije o senzoru, pritišće gumb *Mjerenja*
5. aplikacija prikazuje izbornik za odabir parametara mjerenja, korisnik odabire period i predmet mjerenja, zatim pritišće gumb *Nacrtaj graf*
6. a) za tražene parametre ne postoji upit, aplikacija ispisuje odgovarajuću poruku
   b) aplikacija prikazuje linijski grafikon za tražene parametre

Na slikama (Sl. 4.1) i (Sl. 4.2) prikazano je korištenje filtriranja senzora te odabir parametara za prikaz očitanja senzora.



Slika 4.1 Filtriranje po županijama



4.2 Odabir parametara za prikaz očitanja mjerenja

5. Zaključak

Aplikacija kao produkt Završnog rada objedinjuje teorijski dio zajedno s praktičnim, s razlogom što su specifikacije aplikacije te zadatak Završnog rada trebali biti dobro proučeni, adekvatno podijeljeni na podzadatke, prilagođeni razvojnom okruženju te shodno tome programski oblikovani po standardima modernog razvoja korisničkog sučelja i web-poslužitelja. Aplikacija je testirana u razvojnom i korisničkom načinu što je pokazalo kako aplikacija zadovoljava početne zahtjeve programske potpore koji su definirani zadatkom Završnog rada. Isto tako aplikacija dobro rukuje iznimkama koje se mogu pojaviti tijekom korištenja navedene. Sam softver oblikovan je tako da je čitljiv i pravilno ugniježđen, a time dopušta i dodatne naknadne preinake i dodatke u smislu proširivanja mogućnosti s minimalnim nadopunama po uzoru na već dosad napisani kod.

6. LITERATURA

[1] https://www.influxdata.com/, web stranica baze podataka InfluxDB, (pristupljeno 7.6.2023.)

[2] https://www.postgresql.org/, web stranica baze podataka PostgreSQL, (pristupljeno 7.6.2023.)

[3] http://postgis.net/, web stranica ekstenzije PostGIS, (pristupljeno 7.6.2023.)

[4] https://geojson.org/, web stranica formata GeoJSON (pristupljeno 8.6.2023.)

[5] https://expressjs.com/, web stranica radnog okvira Express (pristupljeno 8.6.2023.)

[6] https://react.dev/, web stranica radnog okvira React (pristupljeno 8.6.2023.)

[7] https://leafletjs.com/, web stranica biblioteke Leaflet.js, pristupljeno (pristupljeno 9.6.2023.)

[8] https://www.chartjs.org/, web stranica biblioteke Chart.js (pristupljeno 9.6.2023.)

7. SAŽETAK

Nadogradnja FER-ove IoT platforme geoprostornim podacima

Problematika Završnog rada bila je proširenje IoT platforme sa geoprostornim aspektom podataka. Napravljena je klijentska aplikacija koja zadanu problematiku rješava uz prikaz karte s raspoređenim senzorima. Podatci o senzorima, poput naziva, lokacije te ID-ja smješteni su u bazi podataka PostgreSQL, a očitanja senzora, tj. mjerenja se povlače iz baze podataka InfluxDB. Čitava aplikacije podijeljena je u 3 podsustava koja međusobnom komunikacijom putem HTTP-a ostvaruju zadani cilj te zadovoljavaju specifikacijske potrebe aplikacije. Ta tri podsustava su *frontend* dio, *backend* dio te baza podataka. Cijela aplikacija napisana je u jeziku Javascript. Korisničko sučelje dizajnirano je u radnom okviru React, dok je poslužitelj pisan u radnom okviru Express. Dodatne biblioteke koje su korištene tijekom razvoja aplikacije su Chart.js i Leaflet.js.

ključne riječi: Internet of Things, senzori, geoprostorni podatci, klijentska aplikacija, baza podataka, korisničko sučelje, aplikacijsko programsko sučelje, HTTP, InfluxDB, PostgreSQL, React, Express

8. Summary

Upgrading FER IoT platform with geospatial dana

The issue we had to adress was to expand FER IoT platform with geospatial data. As the product of this Final paper, the client app has been made. It solves Final assignment with map that contains all FER's sensors. All sensor data is stored in database PostgreSQL, but measurements have been pulled from database InfluxDB. Whole app software is divided into 3 subsystems that comunicate one to each other over HTTP. Those are frontend, backend and database. App is entirely written in programming language Javascript. User interface is designed with framework React, while the web-server is designed with framework Express. External libraries that were used to design this app are Chart.js and Leaflet.js.