Izrada front-end komponenti aplikacije za praćenje otvorenosti školjkaša

Družeta, Nikola

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:793800

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike u Puli

NIKOLA DRUŽETA

IZRADA FRONT-END KOMPONENTI APLIKACIJE ZA PRAĆENJE OTVORENOSTI ŠKOLJKAŠA

Diplomski rad

Pula, rujan, 2021.

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike u Puli

NIKOLA DRUŽETA

IZRADA FRONT-END KOMPONENTI APLIKACIJE ZA PAĆENJE OTVORENOSTI ŠKOLJKAŠA

Diplomski rad

JMBAG: 0303061229, redoviti student

Studijski smjer: Duplomski studij informatike

Predmet: Izrada informatičkih projekata

Znanstveno područje: Društvene zanosti

Znanstveno polje: Informacijske i komunikacijske zanosti

Znanstvena grana: Informatika

Mentor: Doc. dr. sc. Nikola Tanković

Pula, rujan 2021.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za magistra _____ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student_____

U Puli, _____, _____ godin

IZJAVA

o korištenju autorskog djela

Ja, _____ dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom _____ _____ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis_____

Sadržaj

1. Uvod 1

2. Priprema za izradu aplikacije 2

2.1. Početni dogovor 2

2.2. UML dijagram Aplikacije 3

2.3.1. UML dijagram prve inačice aplika 3
2.3.2. UML dijagram druge i teće inačice aplikacije 4
2.3.3. UML dijagram četvrte inačice aplikacije 5

2.3. Odabir tehnologija 6

2.3.1. Vue.js i Vue 3..... 7
2.3.2. Vue CLI..... 8
2.3.3. Vue Router..... 9
2.3.4. NPM..... 9
2.3.5. Apexcharts..... 10
2.3.6. Visual Studio Code..... 11
2.3.7. GitHub..... 12

2.4. Pregledavanje dokumentacije 13

2.5 Dogovor o načinu prenošenja podataka 14

2.5.1. Izgled JSON datoteke prve inačice aplikacije 14
2.5.2. Izgled JSON datoteke druge inačice aplikacije 15
2.5.3. Izgled JSON datoteke treće inačice aplikacije 16
2.5.4. Izgled JSON datoteke četvrte inačice aplikacije 17

3. Razvoj aplikacije 18

3.1. Postavljanje projekta 18

3.2. Implementacija podstranica pomoću router view-a 19

3.3. Komponente 21

3.4. Uvoz i filtriranje podataka 22

3.5. Prosljeđivanje podataka komponentama 23

3.5.1. Izgled JSON datoteke treće inačice aplikacije 23
3.5.2. Izgled JSON datoteke četvrte inačice aplikacije 23

3.6. Pokretanje novog eksperimenta 25

3.7. Grafov 26

3.8. Korisničko sučelje 27

3.9. Obrada podataka. 29

3.9.1. Filtriranje podataka nakon uvoza.....29
3.9.2. Postavljanje statusa
gumba za slanje signala 30
za gašenje eksperimenta
3.9.3. Postavljanje statusa uređaja u tekstualnom obliku..... 31

4. Uputstva za rad ..... 32.

4.1. Početna stranica i izbornik..... 32
4.2. Pokretanje novog eksperimenta..... 33

4.3. Prikaz liste senzora.....34
4.4. Prikaz podstranica Uređaja.....35
4.5. Trenutne greške i nedostatci aplikacije.....36

Zaključak 37

Sažetak 38

Summary 39

Ključne riječi 40

Key words..... 41

Literatura 42

Popis slika..... 43

1. Uvod

Zagađenje je jedan od velikih problema današnjice, a zagađenje mora je jedan od najizraženijih dijelova tog problema. Kako bi se taj problem moglo riješiti treba prvo saznati koji je utjecaj zagađenja na floru i faunu. Ovaj rad dio je projekta pokrenutog točno s tim ciljem, bilježenjem ponašanja školjkaša s obzirom na uvjete u njihovoj okolini. Članovi tima su studenti: Nikola Družeta, Matija Hamer i Kristijan Cetina, dok je doc. dr. sc. Nikola Tanković mentor projekta. Svaki student u timu ima zaseban zadatak, a tema ovog dijela je opis postupka razvoja front end aplikacije u programskom jeziku JavaScript i Vue.js radnom okruženju.

2. Priprema za izradu aplikacije

Prije samog početka izrade aplikacije potrebno je obaviti neke pripreme, na primjer: izrade teorijskog koncepta aplikacije pomoću UML dijagrama, odabir tehnologija koji će se koristiti i pregledavanje dokumentacije tih tehnologija te traženje i pregledavanje besplatnih primjera i tečajeva o korištenju izabranih tehnologija putem interneta.

2.1 Početni dogovor i ideja projekta

Kao što je u uvodu spomenuto, završni cilj ovog projekta je izrada aplikacije za nadzor otvorenosti školjkaša u različitim uvjetima. Na prvom sastanku ustvrđeni su okviri, uvjeti i zahtjevi projekta, kao i podjela posla između članova tima.

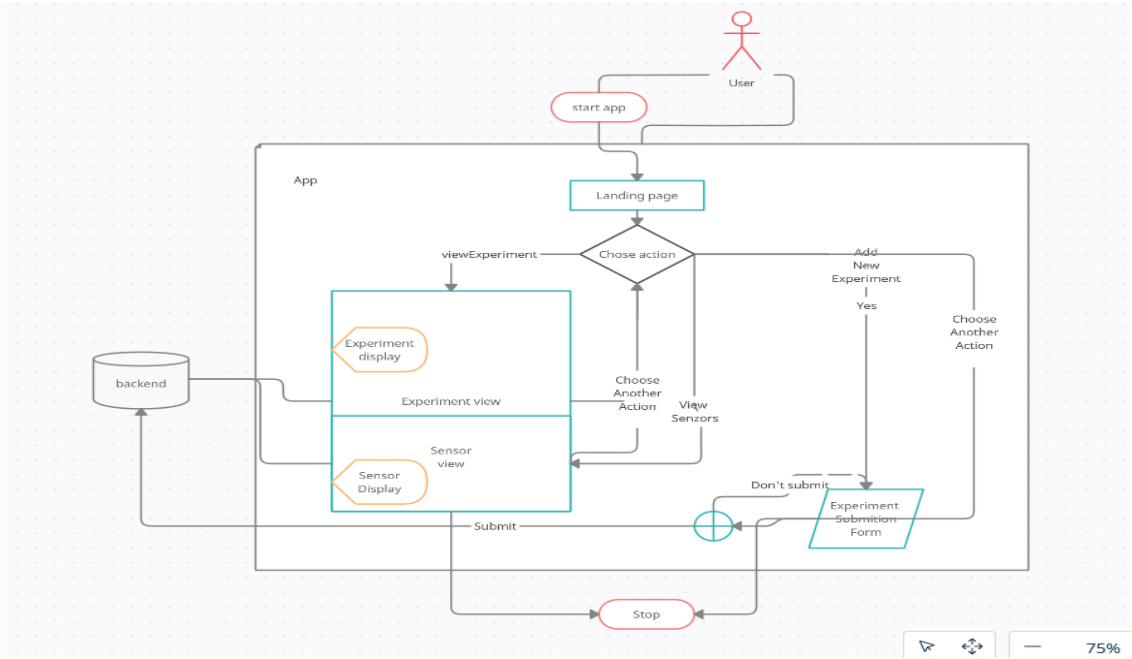
2.2 UML dijagram aplikacije

U svrhu izrade ove aplikacije izrađena su tri UML dijagrama.

2.2.1 UML dijagram prve inačice aplikacije

Prvi je dijagram model prvobitnog prototipa dok je drugi dijagram izrađen naknadno, prije izrade završnog oblika aplikacije. Sa slike 1, koja prikazuje prvi dijagram, vidi se da je početna ideja aplikacije, uz prikaz podataka bila i pokretanje novih pokusa, nakon pokretanja aplikacije dolazilo se na početnu stranicu, s koje se onda moglo birati da li se želi pregledati listu senzora, listu pokusa ili izraditi novi pokus. Vidljivo je također da su senzori bili predviđeni kao podskup prikaza pokusa. Razlog tomu je to da se uz mogućnost prikaza svih prikazati senzore unutar pokusa kojima su oni bili dodijeljeni. Iz ovog dijagrama mogu se iščitati i neki propusti i pogreške u izgradnji modela. Prvi takva greška

Slika1. UML dijagram prve inačice aplikacije



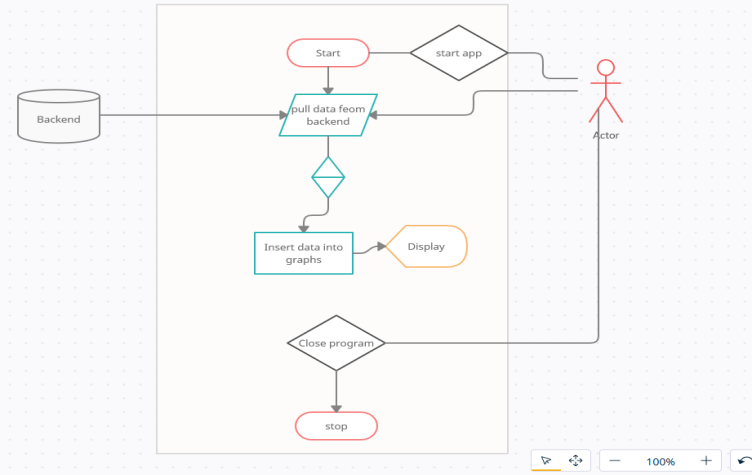
Izvor: Creately

je neurednost modela s vezama koje sijeku tekst te se preklapaju i križaju na načine koje dijagram čine vrlo nerazumljivim te omogućavaju dvosmislenost koja može dovesti do probleme pri kasnijem tumačenju.

2.2.2 UML dijagram druge i treće inačice projekta

Daljnji propusti u izradi dijagrama su nedostatak procesa filtriranja podataka prije njihovog prikaza. Drugi je dijagram u mnogočemu pojednostavljen kao što je vidljivo na slici 2. ovdje veze nisu imenovane ali su uvelike preglednije. Vidljivo je da se uvelike

Slika 2. UML dijagram druge i treće inačice aplikacije



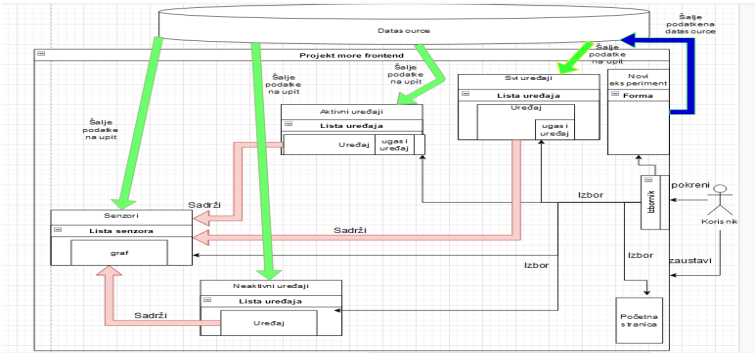
Izvor: Creately

smanjio obujam dijagrame i funkcionalnosti aplikacije. Razlozi ovih drastičnih promjena u odnosu na početni model je ta da se od mogućnosti pokretanja novih pokusa odustalo, dok se zbog promjena u izgledu JSON datoteke izgubila potreba za posebnim prikazima pokusa i senzora. U ovom dijagramu možemo vidjeti da nakon pokretanja aplikacije imamo samo jedan izbor, to jest povlačenje podataka s backenda. Ti podaci se potom filtriraju i prikazuju, nadalje u ovom se dijagramu vidi da je zaustavljanje aplikacije povezano samo s vanjskim korisnikom koji aplikaciju može zaustaviti u svakom trenutku.

2.2.3 UML dijagram završne inačice projekta

Kod daljnjeg rada uvidjelo se da ni ovaj dijagram ne zadovoljava potrebne uvjete. Ponajviše zato što nema mogućnosti prikaza određenih detalja poput statusa uređaja te razlike između kalibracijskih podataka i očitanja tijekom rada kao i nedostatak mogućnosti pokretanja novog eksperimenta te mogućnost gašenja aktivnih eksperimenata. Na sjedećem dijagramu, vidljivom na slici 3, su se ovi problemi uzeli u obzir, što je dovelo do zamršenijeg prikaza više nalik prvotnom, no detaljnije razrađenog. Kao što vidimo front end aplikacija sadrži tri različita prikaza uređaja, prikaz senzora, formu za pokretanje novog eksperimenta te gumba za gašenje aktivnih eksperimenata. Crne strelice nam pokazuju izbore koje korisnik ima na raspolaganju, plava strelica povezuje formu za pokretanje novog eksperimenta sa vanjskim izvorom podataka, koji predstavlja nekakav izvor podataka, bio to backend ili, kao što je to tijekom razvoja bio slučaj JSON datoteka, zelene strelice prikazuju slanje podataka sa vanjskim izvorom podataka u naše prikaze, a crvene strelice nam prikazuju da prikaz senzora može biti dio ostalih prikaza. Iz dijagrama se može iščitati kako korisnik putem izbornika može pristupiti pregledu uređaja, bilo to aktivnim, neaktivnim ili svim uređajima u sustavu, kao i početnoj stranici, i prikazu svim zabilježenih senzora te formi za pokretanje novih eksperimenata. Različite liste uređaja podatke filtriraju na različite načine no njihova struktura je gotovo identična, jedina razlika je da aktivni uređaji, bilo da se prikazuju odvojeno ili zajedno sa neaktivnima imaju gumb za prekid eksperimenta. Podaci koje želimo se povlače sa vanjskog izvora podataka u odabrani pogled te se potom filtriraju i prikazuju.

Slika 3. UML dijagram trenutne inačice aplikacije



Izvor : Flowcharts

2.3.Odabir tehnologije

Nakon izrade UML dijagrama slijedi tehnologija koje ćemo koristiti. Za front end se po savjetu mentora koristio Vue.js radno okruženje. U ovom se slučaju koristiti Vue3 inačicu prije spomenutog radno okruženje. Glavni razlog izbora Vue3 inačice ovog radnog okruženja je to što je pronalazak vizualno-auditivnih uputa za rad s njim bio jednostavniji. U svrhu jednostavnog započinjanja projekta također će se koristiti Vue CLI, namijenjen brzom i jednostavnom postavljanju Vue projekta. Uz to se koristio i Apexcharts.js paket namijenjen izradi grafova izabran zbog jednostavnosti korištenja te kvalitetnog preglednika prikaza grafova. Samo pisanje koda obavljeno je u programu Visual Studio Code izabranom zbog široke potpore rada u različitim programskim jezicima i mogućnosti rada s NPM-om (node packet managerom), koji omogućuje akcije poput dizanja JSON datoteke na lokalnu mrežu tako da se povezivanje na vanjski API može testirati nezavisno o radu drugih članova tima. Osim NPM paketa korišteni su i različiti paketi namijenjeni olakšavanju rada te naglašavanju sintakse u JavaScriptu, HTML-u,CSS-u, JSON-u, Vue.js-u. Treba napomenuti da se uz opće pakete za rad s Vue.js om koristilo i pakete namijenjene radu isključivo sa Vue3.

2.3.1 Vue.js i Vue.3

Radno okruženje Vue.js radno okruženje namijenjeno izradi korisničkih sučelja koje dozvoljava da HTML, Javascript i CSS kod pišemo u istoj datoteci. Ovo je reaktivan alat što znači da se prikaz osvježava svaki pud kad se u Vue datoteci zabilježi nova promjena. To nam omogućuje da promatramo napredak i uočavamo probleme svog koda tijekom samog programiranja.Ova reaktivnost je posljedica deklarativnog učitavanja to jest

Slika 4. prikaz vue koda

Izvor: Codebin

Slika 5. rezultat koda sa slike 4

Hello Vue!!

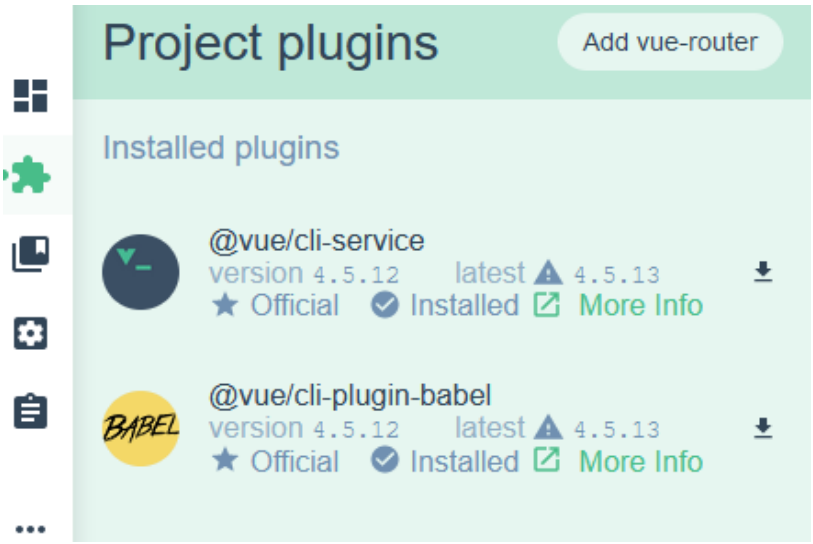
Izvor: Codebin

mogućnosti pisanja JavaScript koda unutar dvostrukih vitičastih zagrada unutar HTML struktura, slike 4. i 5. prikaz su korištenja ove sintakse. U ovom bismo kodu, u JavaScript dijelu trebali promijeniti samo sadržaj varijable message, kako bi se poruka ispisana na ekranu promijenila. Još jedna prednost Vue.js-a su komponente. Pomoću komponenti se velike projekte gleda podijeliti na manje cjeline tako da svaka od tih cjelina bude zastupljena različitom komponentom.Rad s komponentama nam omogućava da pojedine komponente koristimo više puta. Tako na primjer, ukoliko radimo s poljima, možemo za svaki član polja izraditi novu komponentu. Komunikacija između komponenti vrši se tako da nadređena komponenta podređenoj prosljeđuje vrijednosti pri pozivanju podređene komponente unutar svog HTML-a, dok komponente mogu poruke prosljeđivati nadređenima koristeći naredbu emit.

2.3.2. Vue CLI

Vue Cli je alat namijenjen prototipiranju, brzom razvoju Vue aplikacija te postavljanje okvira vue projekta. Ovaj nam alat omogućava da u narednom retku koristimo naredbu “vue”. Ova naredba je ta koja nam omogućuje izvođenje prije spomenutih radnji kao i pokretanje grafičkog sučelja za lakše upravljanje projektom. Ovo se radi dodavanjem

Slika 6. Korisničko sučelje Vue CLI



Izvr Vue CLI ui

različitih nastavaka naredbi Vue, tako ćemo na primjer za pokretanje grafičkog sučelja napisati “vue ui”. Slika 6 prikazuje korisničko sučelje. Kako bi arhitektura Vue CLI bila fleksibilna te imala mogućnost proširivanja ona je zasnovana na priključcima, to jest pluginovima. Priključke se može jednostavno dodavati naredbom “vue add”. Ovo se može napraviti pri započinjanju samog projekta ili naknadno, u trenutku kada primijetimo potrebu za dodavanjem novog paketa. Ukoliko pakete dodajemo nakon što smo uočili potrebu savjetuje se commitanje trenutnog stanja projekta ili neki drugi vid osiguranja trenutne inačice, kako bismo se u slučaju nastanka problema mogli vratiti na stanje prije njegovog nastajanja.

2.3.3. Vue Router

Vue Router je službeni router paket Vue.js-a namijenjen olakšavanju rada s jednostraničnim aplikacijama. Router nam omogućava da jedan HTML objekt jednostavno možemo koristiti za prikaz više komponenata. To se obavlja tako da u JavaScript datoteku uvezeno željene komponente, nakon toga za svaku rutu koju želimo stvoriti moramo definirati putanju, naziv i komponentu koju će ta ruta pozivati. Ukoliko se ukaže potreba za time rute se mogu ugnijezditi, u tome se slučaju ugniježđena ruta vodi kao dijete rute s koje joj se pristupa.

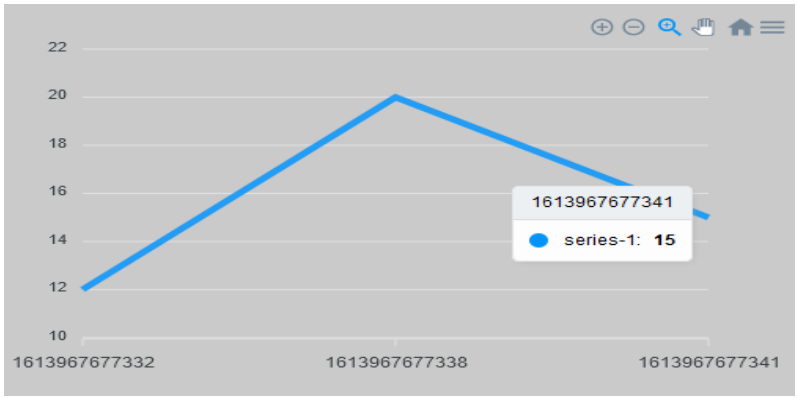
2.3.4. NPM

Node package manager, ili skraćeno NPM je najveći softverski registar na svijetu čiji se paketi korist, dijele i dodaju po open source modelu, NPM je danas temelj velikog broja projekata kao i glavni centar dijeljenja koda u JavaScriptu. NPM se sastoji od tri dijela, Web mjesto na kojem možemo tražiti pakete koji odgovaraju našim potrebama, sučelja naredbenog retka, to jest command line interface (CLI) i registra, te baze podataka dostupnih paketa i njihovih metapodataka.

2.3.5. Apexcharts

Apexcharts je open source projekt stvoren od strane MIT-a. Ova knjižnica nam omogućuje izradu interaktivnih i prilagodljivih dijagrama visoke kvalitete. Uz to grafovi kreirani korištenjem Apexcharta imaju u sebe ugrađenu mogućnost povećavanja i smanjivanja prikaza, micanja lijevo-desno po prikazu te izvoza podataka.Grafovi u Apexchartu također mogu biti dinamični, to jest učitavati podatke s obzirom na izbor dok prikaz tih podataka prate animacije visoke kvalitete.

Slika 7. Primjer grafa izrađenog u apexchartu

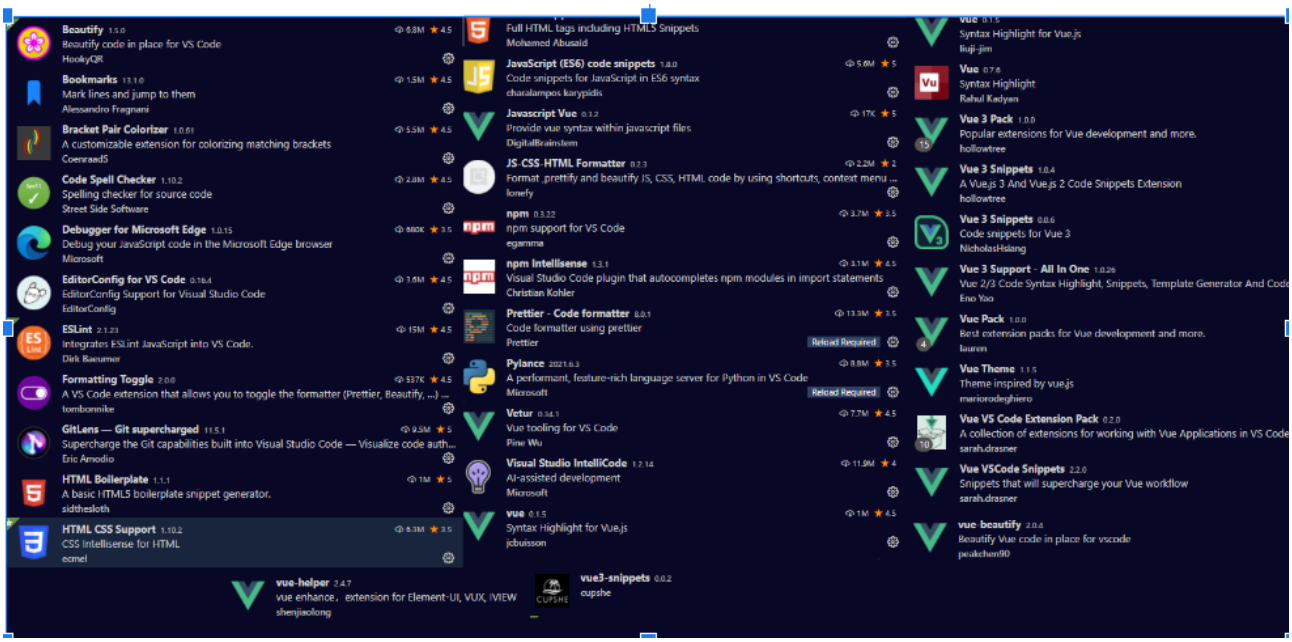


Izvor: Vlastita izrada

2.3.6. Visual Studio Dode

Visual Studio Dode je program namijenjen uređivanju koda izrađen tako da ne zauzima puno prostora za pohranu te da vrši što manje opterećenje na procesor, to ipak ne znači da Visual Studio Code nije svestran i moćan, on pri instalaciji ima ugrađenu podršku za JavaScript, Node.js in TypeScript, no kao sto je vidljivo na slici 8. Visual Studio

Slika 8. Paketi uvezeni U Visaul Studio Code u svrhu izrade ovog projekta



Izvor: Vlastita izrada

Code nudi mogućnost preuzimanja i korištenja velikog broja proširenja za različite programske jezike. Tako putem ovog alata možemo koristiti programske jezike poput C#, C++, Jave, Pythona i mnogih drugih. Osim jezika Visual Studio Code nam omogućava preuzimanje proširenja poput debugger, vizualnih tema te alata za formatiranje izgleda koda koja ubrzavaju rad, čine kod preglednijim te stvaraju ugodnije radno okruženje. Još jedna prednost Visual Studio code-a je ta da je dostupan na Windows, MacOS i Linux platformama, za kraj treba napomenuti da Visual Studio Code nudi mogućnost povezivanja s GitHubom Što nam omogućuje jednostavno dizanje koda na vanjski repozitorij kao i to da Visual Studio Code ima vlastiti naredbeni redak u kojem možemo koristiti NPM naredbe.

2.3.7 GitHub

GitHub je digitalni repozitorij i razvojna platforma koju koristi veliki broj software developera i organizacija. Na GitHubovoj stranici stoji da je ovo najveća i naj naprednija razvojna platforma na svijetu. Ova platforma Služi nam kao host za naš kod te omogućuje nekoliko korisnih operacija. Prva od tih operacije je stvaranje novih grana, to jest stvarati nove inačice bez da mijenjamo prijašnju, ovo se koristi na više načina. Ukoliko na istom kodu radi više ljudi, svatko može izvući svoju granu te raditi na jednom dijelu koda, sve te grane kasnije možemo objediniti bez kolizija i greška u kodu zahvaljujući Git kontroleru. Druga bitna operacija bi bila commit, to jest slanje koda iz programa kao što su Visual Studio Code u repozitorij. Zadnja značajna operacija bila bi pull, to jest povlačenje koja nam omogućava povlačenje koda iz izabrane grane.

2.4. Pregledavanje dokumentacije i primjera

Zadnji korak prije izrade same aplikacije bio je potrebno proučiti dokumentaciju izabranih tehnologija, uz to su se, kao dodatni izvor informacija, koristili razni primjeri dostupni putem internet. Ovaj dio je bio izuzetno bitan zato što ja nisam imao nikakvog iskustva rada s radnim okruženjem Vue i Apexcharts paketom te samu vrlo kratkom vremenu, prateći službenu dokumentaciju i video tečajeve poput Vue JS Crash Course 2021 autora Traversy Media (Traversy Media, 24. 2. 2021.) u vrlo kratkom roku naučio osnove rada s ovim alatima. Ovdje spomenuti video uradak objašnjava sljedeće stavke važne u izradi ove aplikacije: kako povući podatke s udaljenog API-ja, korištenje Vue routera, te uspostavljanje lokalnog backenda korištenjem JSON datoteke, na sličan način je proučen i Apexchart. U ovom slučaju je glavni izvor informacija bila službena stranica Apexchart-a . Zadnji izuzetno bitan izvor informacija bile su web stranice Stack Overflow i Vue Land, službeni Discord server Vue.js radnog okruženja.

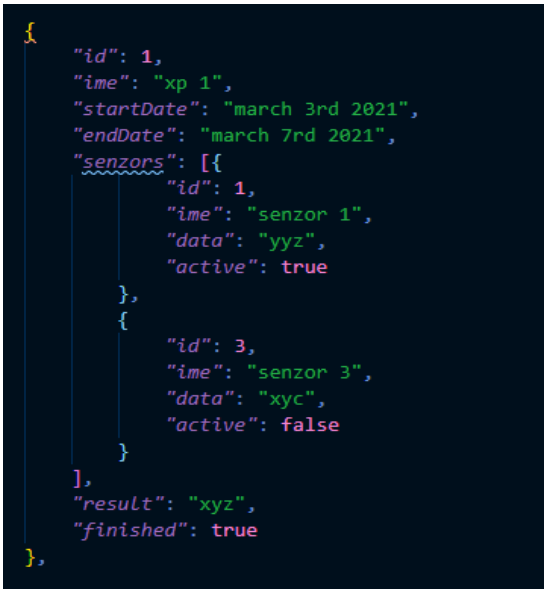
2.5. Dogovor o načinu prenošenja podataka

Članovi tima trebali su se dogovoriti o načinu prenošenja podataka između dijelova projekta, treba napomenuti da se ova etapa ponavljala nekoliko puta. Prvi se put o ovoj temi raspravljalo pri samom pokretanju projekta, tadašnji zaključak je bio taj de će se u svrhu prenošenja podataka koristiti JSON no u tom trenutku nije bila dogovoren nikakav izgled i struktura podataka.

2.5.1 Izgleda JSON datoteke u početnoj inačici aplikacije

Prvi prototip front end aplikacije se bazirao na gruboj pretpostavci prikazanoj na slici 9. U ovom slučaju JSON se sastoji od identifikacijskog broja, imena , datuma početka i kraja

Slika 9. Izgled JSON datoteke prve inačice aplikacije



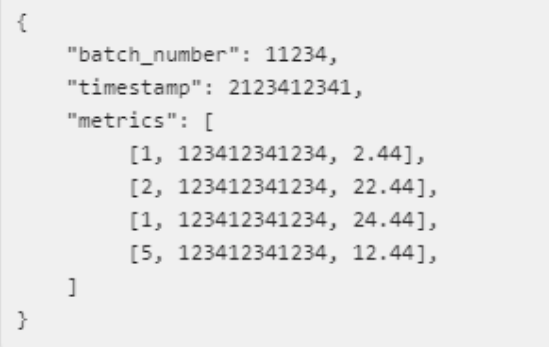
Izvor: Vlastita izrada

pokusa te statusa pokusa s obzirom na to je li experiment završio ili je još u tijeku. Senzori su bili ugniježđeni unutar pokusa no treba napomenuti da se koristila i druga JSON datoteka namijenjena isključivo senzorima.

2.5.2 Izgleda JSON datoteke u drugoj inačici aplikacije

Prvotna ideja je bila implementirati nekoliko različitih načina filtriranja pokusa i senzora od kojih se kasnije odustalo, što je izbacilo potrebu za velikim brojem varijabli iz prvotnog primjera, zato je tijekom naknadnih razgovora na ovu temu mentor projekta predložio izgled JSON podataka prikazan na slici 10. Vidljivo je da je ovaj ustroj podataka puno jednostavniji. Ovdje se u potpunosti briše potrebu za stvaranjem dvije posebne liste te

Slika 10. Izgled JSON datoteke druge inačice aplikacije



Izvor: Vlastita izrada

prosljeđuje broj pošiljke koji služi kao identifikacijski broj pošiljke podataka, vremensku oznaku pošiljke i metričke podatke u obliku ugniježđene liste. U toj listi se svaki vanjski član sastoji od tri unutarnja gdje prvi unutarnji član označava identifikacijski broj senzora, drugi vremensku oznaku očitanja a treći vrijednost koju je senzor očitao.

2.5.3 Izgleda JSON datoteke u trećoj inačici aplikacije

Uz daljnje pojednostavljivanje dolazi se do trećeg izgleda JSON podataka prikazanom na slici 11. ovaj paket sadrži samo vremensku oznaku te listu podataka očitanja pojedinih senzora. Prema objašnjenju jednog člana tima, ovaj oblik podataka je izabran kako bi se

Slika 11. Prikaz JSON datoteke treće inačice aplikacije

```
[{
    "timestamp": 738127638,
    "data":[{
            "senzor1": 525,
            "senzor2": 625,
            "senzor3": 425,
            "senzor4": 325
    }]
}
]
```

Izvor: Vlastita izrada

poklapao sa standardnim praksama korištenja JSON-a. U ovom je dijelu planiranja napravljen veliki propust jer ovakav oblik JSON-a ne osigurava kvalitetan i cjelokupni pregleda podataka bitnih za eksperimente koje provodimo.

2.5.4. izgleda JSON datoteke u četvrtoj inačici aplikacije

Kod četvrte inačice izgleda podataka u JSON-u odlučilo se dodati puno veći broj podataka vidljivom na slici 12. Kao što sa slike možemo vidjeti senzori su zabilježeni kao poseban objekt ,te kao dio objekta uređaja što nam omogućuje da ih promatramo nezavisno od uređaja kojima pripadaju ili kao dio promatranog uređaja. Sami senzori sadrže četiri liste podataka, dvije za vremenske oznake očitanja i dvije za podatke. Ovaj oblik je izabran zato da senzor osim podataka i vremenskih oznaka mjerenja može jednostavno pohraniti i podatke i vremenske oznake kalibracije. U koju će se listu upisati koji podatak nadzirat će backend koji vodi računa dobiva li podatke kalibracije ili stvarnog mjerenja. Uređaji uz podatke o svojim senzorima također čuvaju podatke o vremenu početka svog zadnjeg eksperimenta, vremenu njegovog kraja, to jest zakazana kraja ukoliko eksperiment još traje te ima zakazano vrijeme završetka, status kalibracije the status aktivnosti. Kalibracija i aktivnost su obilježeni zasebno kako bi bilo lakše prikazati u kojoj fazi rada je uređaj trenutno, ovo će biti detaljnije razjašnjeno kod opisa koda.

Slika 12. Prikaz JOSN datoteke završne inačice aplikacije


```
"uredjaji": [{
            "id": 1,
            "startDate": "03 03 2021 13:49:51.141",
            "endDate": "",
            "kalibracija": false,
            "active": true,
            "senzori": [{
                    "id": 1,
                    "timestamp": [
                        1613967677332,1613967677338,1613967677341
                    ],
                    "calibrtionTimestamp": [
                        1213967677231,1213967677237,1213967677240
                    ],
                    "calibrationData": [
                        0,1, 0
                    ],
                    "data": [
                        12,20,15
                    ]
                }
            }]
    "senzori": [{
            "id": 1,
                    "timestamp": [
                        1613967677332,1613967677338,1613967677341
                    ],
                    "calibrtionTimestamp": [
                        1213967677231,1213967677237,1213967677240
                    ],
                    "calibrationData": [
                        0,1, 0
                    ],
                    "data": [
                        12,20,15
                    ]}
    ]
```


Izvor: Vlastita izrada

3. Razvoj aplikacije

3.1. Postavljanje projekta

Izrada projekta započela se preuzimanjem i instaliranjem Vue CLI putem NPM-a. Nakon toga se konzola naredbenog retka postavila na željeni direktorij te se pomoću prije instaliranog Vue CLI-a postavlja sam projekt. Kod postavljanja projekta smo uz zadane postavke dodalo i Router svojstvo, koje olakšava rad s većim brojem podstranica.,sljedeći korak je bio uspostava lažnog backenda, to jest podizanje JSON datoteke na lokalnog poslužitelja kako bi se u aplikaciji moglo isprobati povezivanje s vanjskim API-jem. Kako bismo to omogućili trebalo je u datoteku package.JSON , pod “script” dodati novu naredbu koju kasnije možemo pozvati putem NPM-a. Treba napomenuti da u ovoj naredbi moramo postaviti koju datoteku će naredba koristiti, te na kojem portu će ona biti dostupna. Izgled te naredbe prikazan je na slici 13. Sa slike je vidljivo da se za podatke koristi datoteka db.JSON , a kao port se koristi port na localhost:5000. Sljedeći korak je bio dodavanje svih pogleda i komponenti koji će biti potrebni za kvalitetan rad aplikacije.

Slika13. implementacija narebe NPM run backend

```
"scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "backend": "json-server --watch db.json --port 5000"
},
```

Izvor: Vlastita izrada

3.2. implementacija podstranica pomoću router view-a

Podstranice nam omogućavaju da podatke jednostavno i uredno podijelimo i prikažemo te se one u Vue-u implementiraju pomoću Vue routera koji podstranice obilježava kao poglede (view), treba naglasiti da su pogledi i komponente dvije zasebne stvari te se spremaju u posebne datoteke. Kako bismo implementirali vue router potrebno je izvesti nekoliko radnji, za početak u glavnu Vue datoteku, App.Vue, unutar HTML template-a dodati router view kao što je to napravljeno na slici 14., nakon toga potrebno je

Slika 14. Implementacija Vue routera

```
import { createRouter, createWebHistory } from 'vue-router'
import Home from '../views/Home'
import ExperimentView from '../views/ExperimentView'

const routes = [{
        path: '/',
        name: 'Home',
        component: Home
    },
    {
        path: '/experimenti',
        name: 'ExperimentView',
        component: ExperimentView
    },
]

const router = createRouter({
    history: createWebHistory(process.env.BASE_URL),
    routes,
})

export default router
```

Izvor: Vlastita izrada

definirati različite rute što činimo u datoteci index.js., to obavljamo tako da u datoteku uvezemo sve poglede te za svaku rutu definiramo stazu (path), naziv i komponentu, to jest view, kao što je prikazano na slici, na kraju moramo omogućiti prijelaz između ruta što je

Slika 15. Implementa router-link poveznica u izbornik

```
<template>
  <div>
    <br/>

    <router-link class="btn"  to="/SviUredjaji">Svi uređaji</router-link>
    <router-link class="btn"  to="/AktivniUredjaji">Aktivni uređaji</router-link>
    <router-link class="btn"  to="/ProslaMjerenja">Prosla mjerenja</router-link>
    <router-link class="btn"  to="/Senzori">Lista senzora</router-link>
    <router-link class="btn"  to="/NoviExperiment">NoviExperiment</router-link>
    <router-link class="btn"  to="/">Početna stranica</router-link>

  </div>
</template>
```

Izvor: Vlastita izrada

kod ove aplikacije napravljeno u komponenti Menu, u čiji smo HTML template dodali router link oznake za svaku dostupnu rutu kao što je to prikazano na slici 15. Ukoliko usporedimo broj ruta na slikama možemo vidjeti da na slici 14 i slici 15 imamo različiti broj ruta, razlog tome je zato što su ove slike uzete iz različitih inačica aplikacije, naime zbog različitih oblika JSON datoteke u svakoj inačici aplikacije drastično se mijenjala i struktura aplikacije. Tako su u prvotnoj inačici bile potrebna tri pogleda: Naslovna stranica, Eksperimenti i senzori, kod sljedeće su inačice ostale samo dvije rute: Naslovna stranica i eksperimenti, dok je kod trenutne inačice, zbog velikog povećanja opsega mogućnosti, potrebno šest ruta koje ćemo razjasniti kasnije. Pogledi nam dopuštaju da koristeći iste komponente prikažemo drugačije filtrirane podatke tako što u svaki pogled umetnemo način filtriranja podataka potreban za obavljanje njegove zadaće, te profiltrirane podatke proslijedimo komponentama. Prikaz koda jednog Pogleda vidljiv je na slikama 16, 17 i 18. Dok je lista svih pogleda vidljiva na slici 19.

Slika16. prvi dio koda pogleda "SviUredjajiView"

```
<template>
  <div>
    <UredjajList :uredjaji="uredjaji">
    </UredjajList>
  </div>
</template>
```

Izvor: Vlastita izrada

Slika17. Drugi dio koda pogleda "SviUredjajiView"

```
<script>
import UredjajList from '@/components/UredjajList'
export default {
    name:'SviUredjaji',
    components:{
        UredjajList
    },
    data(){
      return{
        uredjaji:[]
      }
    },
```

Izvor: Vlastita izrada

Slika 18. Treći dio koda pogleda "SviUredjajiView"

```
    methods:{
      async fetchUredjaji(){
            const res=await fetch('api/uredjaji')

            const data= await res.json()
            return data
        },
    },
    async created() {
      this.uredjaji=await this.fetchUredjaji()
    }

}
</script>
```

Izvor: Vlastita izrada

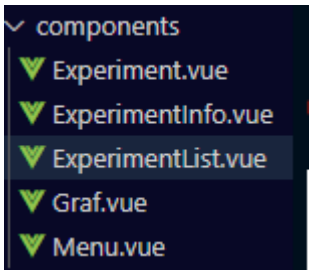
Slika19. Lista pogleda u završnoj inačici aplikacije

Izvor: Vlastita izrada

3.3. Komponente

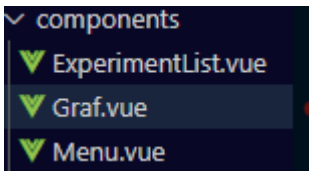
Kao što je prije naglašeno Pogledi i komponente su dvije zasebne cjeline. Pogledi služi izvršavanju nekog zadatka kao što je prikaz određene grupe podataka ili pokretanje novog eksperimenta, dok su komponente te koje nam omogućavaju izvršavanje tih zadataka. Tako na primjer pogled “SviUredjaji” u završnoj inačici u sebi poziva komponentu “UredjajList” koji pak pomoću petlje za svaki uređaj na listi poziva novu komponentu tipa “Uredjaj” te u nju sprema podatke o jednom uređaju. Komponenta “Uredjaj” osim komponente “SenzorList” u sebi poziva i Komponente “ShowSenzorList” te “SendShutdownSignal” koje, kao što im ime sugerira, omogućavaju da listu senzora prikažemo i minimiziramo po izboru te da, ukoliko se radi o aktivnom uređaju, na datasource pošaljemo signal za njegovo gašenje. Ovdje je očito da unutar jednog pogleda ili čak jedne komponente nismo ograničeni na određen broj komponenti koje možemo pozvati, bilo da se petljom stvara više inačica iste ili da se ručno dodaje više različitih komponenti. Očito je da su nam, kao i kod pogleda, kod različitih inačica aplikacije bile potrebne različite komponente. U trenutnoj, to jest četvrtoj, inačici aplikacije, na primjer imamo dvanaest komponenti, dok je treća inačica imala tri, a druga pet. Prikaz tih komponenti vidljiv je na slikama 20, 21 i 22.

Slika 20. Komponente druge inačice aplikacije



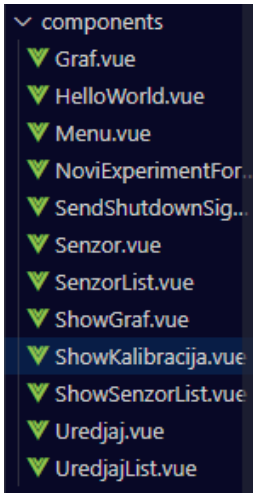
Izvor: Vlastita izrada

Slika 21. Komponente treće inačice aplikacije



Izvor: Vlastita izrada

Slika 22. Komponente završne inačice aplikacije



Izvor: Vlastita izrada

3.4. Uvoz podataka

Kako bismo sa našeg lažnog backenda dohvatili podatke korišten je fetch API. koji zaprima web adresu na kojoj se podaci nalaze te se sa te adrese povlači podatke u aplikaciju. Naši podatci se nalaze na adresi “localhost:5000” no kako bi se olakšao proces eventualne izmjene adrese s koje se podaci povlače u projekt je dodana datoteka vue.config.js u kojoj se “api” definira kao zamjena za pisanje adrese. Ovo se napravilo zato što se fetch funkcija poziva više puta te uvijek pristupa drugoj podstranici adrese

Slika 23. definiranje api kratice

```
module.exports = {
    devServer: {
        proxy: {
            '^/api': {
                target: 'http://localhost:5000',
                changeOrigin: true,
                logLevel: 'debug',
                pathRewrite: { '^/api': '/' }
            },
        }
    }
}
```

Izvor: Vlastita izrada

Slika 24. Naredba Fetch

```
methods:{
  async fetchSenzori(){
      const res=await fetch('api/senzori')

      const data= await res.json()
      return data
  },
},

async created() {
    this.senzori=await this.fetchSenzori()
}
```

Izvor: Vlastita izrada

localhost:5000. Prikaz naredbe fetch the definicija “api” vidljive su na slikama 23 i 24. Bitno je napomenuti da funkcija za dovlačenje podataka koristeći fetch mora biti asinkrona, asinkronost nam omogućuje da funkcija vrati obećanje umjesto stvarnih podataka koji se dostavljaju naknadno što je vrlo korisno pri povlačenju podataka s mreže. Ukoliko je funkcija asinkrona pri njenom pozivanju moramo koristiti naredbu “await” koja označava da čekamo njeno izvršenje, to je prikazano u operaciji “created” vidljivoj na slici 24. Nakon što smo podatke prihvatili možemo ih ispisati na ekranu. Ovi podaci su nefiltrirani te teško čitljivi te prikazani na slici 25. Filtriranje podataka se naravno razlikuje ovisno o inačici, no u posljednjoj inačici aplikacije filtriranje se vrši jednostavnom for petljom koja provjerava parametre kalibracije i aktivnosti uređaja. Kod ove petlje biti će prikazan kasnije u ovom radu.

Slika 25 Prikaz nefiltriranih i neuređenih podataka

[{ "batch_number": 11234, "timestamp": 2123412341, "metrics": [[1, 123412341234, 2.44], [2, 123412341235, 22.44], [3, 123412341234, 24.44], [4, 123412341234, 12.44]] }, { "batch_number": 11235, "timestamp": 2123418902, "metrics": [[1, 123912341270, 23.44], [2, 123412381273, 2.44], [5, 123412377271, 34.44], [4, 123912341270, 19.44]] }, { "batch_number": 11236, "timestamp": 2123419987, "metrics": [[1, 123912341278, 82.44], [5, 123912341275, 29.44], [2, 123912341275, 28.44], [3, 123912341280, 54.44], [4, 123912341271, 10.44]] }]

[[[1, 123412341234, 2.44], [1, 123912341270, 23.44], [1, 123912341278, 82.44]], [[2, 123412341235, 22.44], [2, 123412381273, 2.44], [2, 123912341275, 28.44]], [[3, 123412341234, 24.44], [3, 123912341280, 54.44]], [[4, 123412341234, 12.44], [4, 123912341270, 19.44], [4, 123912341271, 10.44]], [[5, 123412377271, 34.44], [5, 123912341275, 29.44]]]

Izvor: Vlastita izrada

3.5. Prosljeđivanje podataka komponentama

3.5.1 Prosljeđivanje u pozvanu komponentu

Nakon uvoza kojeg smo obradili te obrade podataka koju ćemo prikazati kasnije, podatke prosljeđujemo komponentama kojima su one potrebne, treba ponovno napomenuti da više pogleda koristi iste komponente, na primjer: pogledi “SviUredjaji”, “AktivniUredjaji” i “ProslaMjerenja” podatke prosljeđuju komponenti “UredjajList”. Prosljeđivanje podataka vrši se putem propertija tako da se kod pozivanja komponente unutar pogleda ili druge komponente upiše ime koje smo propertiju dali u komponenti koju pozivamo te u njega spremimo vrijednost koju želimo proslijediti. Prikaz prosljeđivanja vidljiv je na slici 26. u komponenti propertije definiramo naredbom “props” prikazanoj na slici

Slika 26. Prikaz prosljeđivanja parametara

```
<div>

  <UredjajList :uredjaji="uredjaji">
  </UredjajList>


</div>
```

Izvor: Vlastita izrada

27. Sa te slike također možemo iščitati i da propertiju pri definiranju treba dodati tip. Prosljeđivanje se može ukomponirati u petlju što je već bilo napomenuto u prijašnjim

Slika 27 Primanje uređaja prosljeđenih

```
props: {

    uredjaji:Array


},
```

u komponentu

Izvor: Vlastita izrada

Slika 28. Primjer v-for petlje i prosljeđivanja zasebnih elemenata polja

```
<div    :key="uredjaj.id" v-for="uredjaj in uredjaji">
  <Uredjaj   :uredjaj="uredjaj"/>
</div>
```

Izvor: Vlastita izrada

poglavljima. Primjer toga je vidljiv na slici 28. Ovaj primjer prikazuje kako se unutar komponente “UredjajList” svaki član liste “uredjaji” prosljeđuje u zasebni poziv komponente “Uredjaj”. Ovdje petljom “v-for” prolazimo kroz svaki član liste “uredjaji” ovdje označen kao “uredjaj in uredjaji” dok nam ključ, u ovom slučaju “uredjaj.id” služi kao osiguranje unutarnje strukture komponente.

3.5.2 Prosljeđivanje iz pozvane komponente u pozivatelja

Prosljeđivanje se može vršiti i u obrnutom smjeru. Tako na primjer komponenta "SendShutdownSignal" pomoću naredbe "emit" šalje podatke u komponentu "Uredjaj" koja je poziva. Emit prima dvije vrijednost: ime i objekt kojeg prosljeđuje, ovo je prikazano na

Slika 29. Prikaz naredbe emit

```
onClick() {

  const newShotdown= {

    uredjajID:this.uredjajID
  }
  this.$emit('shutdown-signal',newShotdown)
},
```

Izvor: Vlastita izrada

slici 29. U komponenti "uredjaj" se signal prima na način prikazan na slici 30, ovdje je vidljivo da komponenta "SendShutdownSignal" prima vrijednost "uredjajID" te vraća istu enkapsuliranu u objekt, ovo se zbog jednostavnijeg prosljeđivanja objekta na datasource.

Slika 30. Primanje emitanog objekta

```
<div v-show="isActive()">
  <SendShutdownSignal :uredjajID="uredjajID" @shutdown-signal="sendEndSignal"></SendShutdownSignal>

</div>
```

Izvor: Vlastita izrada

Prosljeđivanje se obavlja koristeći naredbu fetch, za razliku od povlačenja podataka koristeći naredbu fetch kod slanja podataka na datasource podataka moramo postaviti metodu naredbe na "POST" umjesto zadane "GET", koju koristimo za povlačenje podataka. Primjer naredbe fetch za slanje podataka an datasource vidljiv je na slici 31. Na slici je vidljivo da objekt kojeg šaljemo na datasource moramo pretvoriti u string kako bismo ga mogli proslijediti u datasource.

Slika 31. Prosljeđivanje podataka na datasource

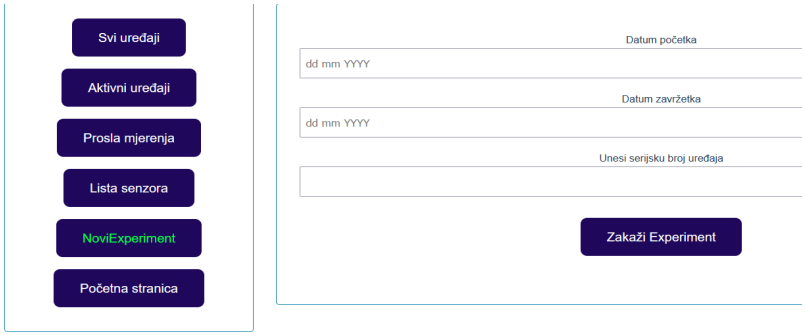
```
async sendEndSignal(end){
  const res=await fetch('api/stopsignal',{
    method:'POST',
    headers:{
      'Content-type': 'application/json',
    },

    body: JSON.stringify(end),
  })

    const data=await res.json()
    this.stopsignal=[...this.stopsignal,data]

}
```

Izvor: Vlastita izrada

3.6. Pokretanje novog eksperimenta

Pokretanje novog eksperimenta bilo je predviđeno u početnoj inačici aplikacije no bilo je izbačeno u naknadnim inačicama. Nakon što se ovu odluku uvidjelo kao grešku odlučeno je u trenutnu inačicu aplikacije vratiti ovu mogućnost. Pokretanje novog eksperimenta odvija se u zasebnom pogledu koji poziva komponentu u kojoj je definirana

Slika 32. Prikaz forme za pokretanje novih eksperimenata u grafičkom sučelju



Izvor: Vlastita izrada

Slika 33. Prikaz koda forme za pokretanje novih eksperimenata

```
<form @submit="onSubmit" class="add-form">
    <div class="form-control">
    <label>Datum početka</label>
    <input
    type="text" name="datumStart"
     placeholder="dd mm YYYY" v-model="datumStart"  required/>
    </div>

    <div class="form-control">
    <label>Datum završetka</label>
    <input
    type="text" name="datumEnd" v-model="datumEnd"
     placeholder="dd mm YYYY"/>
    </div>

    <div class="form-control">
    <label>Unesi serijsku broj uređaja</label>
    <input
      type="number" name="broj" v-model="broj"  required/>
    </div>

    <input type="submit" value="Zakaži Experiment"
    class="btn"/>
</form>
```

Izvor: Vlastita izrada

forma, izgled the forme vidljiv je na slici 32 dok je na slici 33 vidljiv kod te forme. Ovdje možemo uočiti da forma prima tri podatka, datum početka eksperimenta, datum završetka the ID uređaja koji će biti korišten, uz to da su datum početka i ID uređaja zabilježeni kao obavezni, dok je datum završetka opcionalan. Ovi podaci se šalju na datasource na isti način kao i podaci za gašenje eksperimenta.

3.7. Grafovi

Za izradu grafova u ovoj aplikaciji korišten je Apexchart koji omogućava jednostavno kreiranje interaktivnih i preglednih grafova na slikama 34 i 35 vidimo kod jednog Apexchart grafa dok na slici 36 vidimo izgled grafa proizašlog iz tog koda, iz slike vidimo da se graf uvozi i poziva kao komponenta kojoj moramo proslijediti četiri parametra: širinu prikaza grafa, tip grafa, postavke grafa i seriju podataka. U postavke grafa spadaju ID grafa te vrijednosti osi grafa, kao što sa slike vidimo da se seriju podataka kao i postavke osi definira kao prazan skup. Ukoliko bi se ovdje pokušalo podatke u graf dodati izravno u seriju podataka, to jest, postavke osi, iz tog razloga se podatke u graf dodaje naredbom “created” koja puni nama potrebna lista podataka pri kreiranju sa podacima koje smo proslijedili komponenti iz koje pozivamo graf.

Slika 34. Prvi dio koda Apexchart grafa

```
<template>
  <div >
    <apexchart
     width="500"
     type="line"
     :options="chartOptions"
     :series="series"
    ></apexchart>


  </div>
</template>

<script>
import VueApexCharts from "vue3-apexcharts";
export default {
    name:'Graf',
    components: {

    apexchart: VueApexCharts,

  },
```

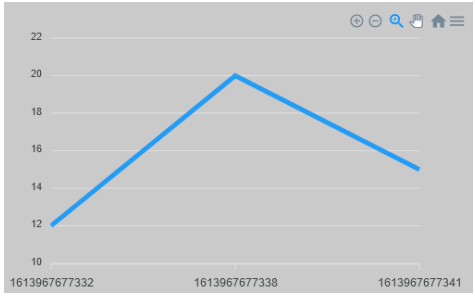
Izvor: Vlastita izrada

Slika 35. Drugi dio koda Apexchart grafa

```
data: function() {
  return {

    chartOptions: {
      chart: {
        id: "graf",
      },
      xaxis: {
        categories: [],
      },
    },
    series: [
      {
        name: "series",
        data: [],
      },
    ]
    };
  },
  props:{
       data: Array,
       timestamp:Array
   },
  async created() {
    this.chartOptions.xaxis.categories=this.timestamp
    this.series=[{data: this.data}]
  }
}
```

Izvor: Vlastita izrada

Slika 36. Graf izrađen kodom prikazanom na slikama 34 i 35

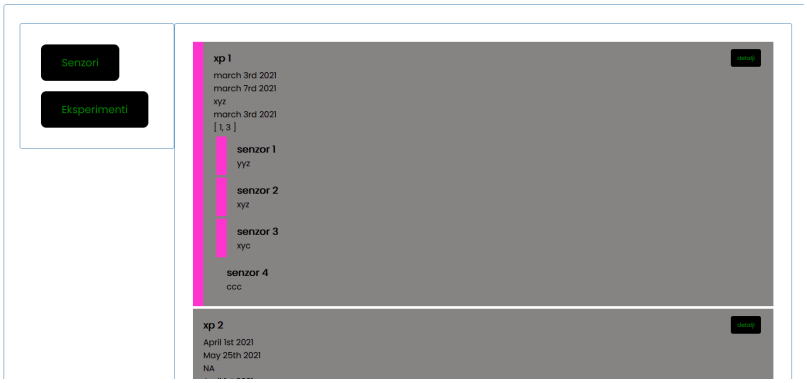


Izvor: Vlastita izrada

3.8. Korisničko sučelje

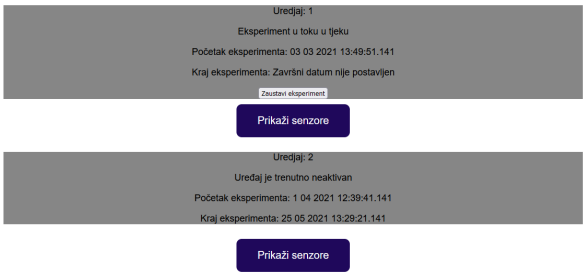
Korisničko sučelje Frontend aplikacije Projekta more je vrlo jednostavno te je njegov koncept prošao najmanje promjena. Razlike između početnog i trenutnog prikaza prikazane su na slikama 37, 38 i 39. Korisničko sučelje sastoji se od dva dijela: Menu-a,

Slika 37. Prikaz korisničkog sučelja prve inačice aplikacije



Izvor: Vlastita izrada

Slika 38. Prikaz korisničkog sučelja završne inačice aplikacije sa sakrivenom listom senzora



Izvor: Vlastita izrada

Slika 39. Prikaz korisničkog sučelja završne inačice aplikacije sa otkrivenom listom senzora

Izvor: Vlastita izrada

namijenjenog promjeni podstranice ,te route view dijela namijenjenog prikazu komponenti specifičnih za podstranicu. Menu se sastoji od router linkova na svaku dostupnu podstranicu. Pomoću klase “btn” definirane u “style” djelu datoteka “App.vue” iz koje se pozivaju pogledi i komponente. U prikazu podstranica izgled se povlači iz pogleda Koji pak povlače izgled definiran u komponentama. Ovo čini izgled većine podstranica u trenutnoj inačici aplikacije veoma sličnim, pošto se u pravilu iste komponente koriste u više pogleda. Razlike su primjetne kod pogleda za pokretanje novih eksperimenata, početne stranice te pogleda namijenjena prikazu senzora. Na slikama možemo vidjeti izgled pogled, a koji prikazuje sve uređaje, ovdje na slici jasno vidimo rezultate pozivanja komponente “Uredjaj” pomoću petlje, dok na slici vidimo slučaj pozivanja komponente “Senzor” unutar komponente “Uredjaj”.

3.9. Obrada podataka.

3.9.1 Filtriranje podataka nakon uvoza

U ovom je radu više puta spomenuta obrada podataka, prvi korak u obradi nakon uvoza podataka jest filtriranje podataka koje iz pogleda prosljeđujemo u komponentu, na slici 40. je prikazan jedan primjer filtriranja podataka iz završne inačice aplikacije.

Slika 40. Petlja koja neaktivne uređaje sprema u varijablu "temp"

```
for (let i = 0; i < this.uredjaji.length; i++) {
  if( this.uredjaji[i].kalibracija==false && this.uredjaji[i].active==false ){
    this.temp.push(this.uredjaji[i])
  }
}
this.uredjaji=this.temp
```

Izvor: Vlastita izrada

For petlja prikazana na slici nalazi se u pogledu “ProslaMjerenjaView” te vraća nam sve neaktivne uređaje u listi, to čini tako što prolazi kroz listu uređaja te ukoliko i-ti uređaj ima vrijednosti “kalibracija” i “active” postavljene na “false” tu vrijednost vraća u listu “temp” Nakon što prođemo kroz cijelu petlju, elemente liste temp spremamo u listu “uredjaji”, ovaj korak nije neophodan te bi se komponenti bez greške mogla proslijediti lista “temp”, no zbog jednostavnosti razumijevanja i konzistentnosti nazivlja izabran je trenutni pristup. Ovakvo filtriranje podataka u pogledima “SenzoriView” i “SviUredjajiView” nije potrebna zato što pogled "SenzoriView" radi za drugačijim setom podataka, dok je svrha pogleda "SviUredjajiView" ta da prikazuje sve uređaje neovisno o njihovom statusu. Dok petlja za filtriranje aktivnih uređaja u pogledu “AktivniUredjajiView” radi na vrlo sličan način, no umjesto spremanja podataka u privremenu listu ona jednostavno briše sve neaktivne podatke iz liste “uredjaji”.

3.9.2 Postavljanje statusa gumba za slanje signala za gašenje eksperimenta

Sljedeći korak je da podatke iz svih prije navedenih pogleda, izuzev “SenzoriView” pošaljemo u komponentu “UredjajList” koja na prije opisani način podatke prosljeđuje u komponentu “Uredjaj”. U komponenti “Uredjaj” se u vezi obrade podataka obavljaju dvije zadaće: provjera aktivnosti uređaja, te postavljanje statusa uređaja u tekstualnom obliku. Na slici 41 prikazan je kod provjere aktivnosti koji određuje hoće li se na tom uređaju

Slika 41. Kod metode provjere aktivnosti uređaja

```
isActive(){
  if(this.uredjaj.kalibracija==true && this.uredjaj.active==false){
    return true
  }else if(this.uredjaj.kalibracija==false && this.uredjaj.active==true){
    return true
  }else if(this.uredjaj.kalibracija==false && this.uredjaj.active==false){
    return false
  }else{
    return true
  }
},
```

Izvor: Vlastita izrada

aktivirati gumb za slanje signala za prekid eksperimenta. Ovdje se radi o if-else uvjetima iz kojih možemo iščitati da se gumb za slanje signala za gašenje aktivira u slučajevima kad je svojstvo “kalibracija” true a svojstvo “active” false, ili ako je vrijednost “kalibracija” false, a svojstvo “active” true, to jest ako je uređaj aktivan ili se kalibrira. Ukoliko su oba ova svojstva false, to znači da je uređaj neaktivan te metoda “isActive” u ovom slučaju vraća false, naknadno je na kraj metode dodano, ukoliko se ne desi ni jedan od navedenih događaja, da metoda vraća vrijednost true. Ovo ima dvojaku svrhu, prva je izbjegavanje greški kod nepredviđenih vrijednosti, a druga je mogućnost gašenja eksperimenata koji rade neispravno.

3.9.3 Postavljanje statusa uređaja u tekstualnom obliku

Na slici 42. je prikazan kod za postavljanje status uređaja u tekstualnom obliku. Kod ove metode sličan je kodu metode za provjeru aktivnosti s tom razlikom da umjesto vraćanja

Slika 42 Kod metode koja postavlja statusa uređaja u tekstualnom obliku

```
setStatus(){
  var stat
  if(this.uredjaj.kalibracija==true && this.uredjaj.active==false){
    stat="Kalibracija u tjeku"
  }else if(this.uredjaj.kalibracija==false && this.uredjaj.active==true){
    stat="Eksperiment u tjeku"
  }else if(this.uredjaj.kalibracija==false && this.uredjaj.active==false){
    stat="Uređaj je trenutno neaktivan"
  }else{
    stat="NEISPRAVNO OČITANJE"
  }

  return stat
},
```

Izvor: Vlastita izrada

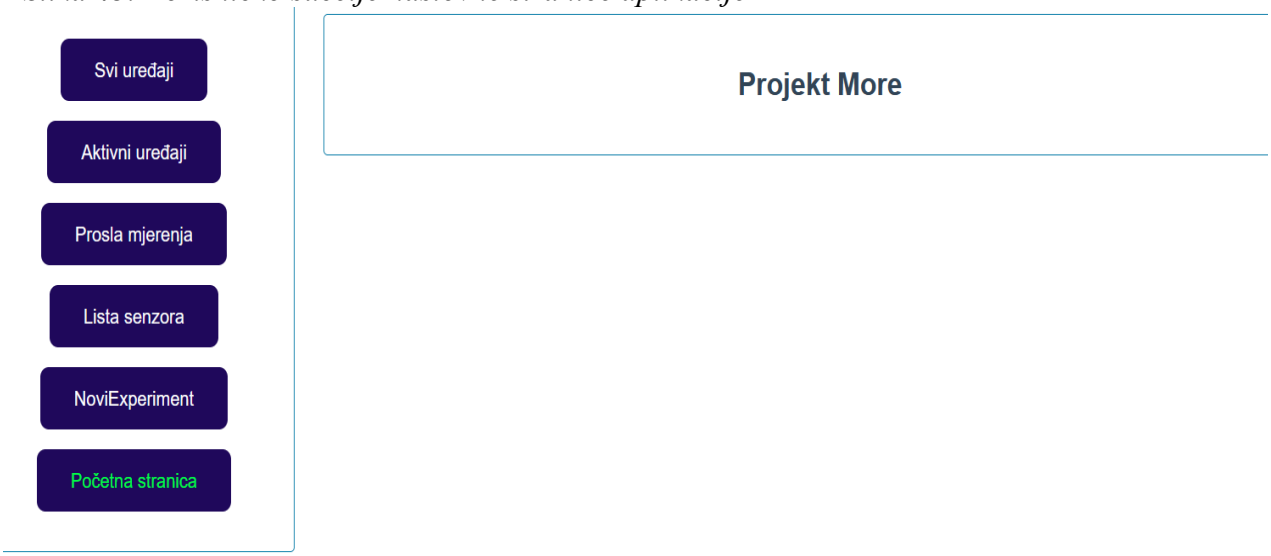
boolean vrijednosti ova metoda sprema string u varijablu “stat”, te vraća vrijednost te varijable. Kao što sa slike vidimo varijabla “stat” ima četiri moguće vrijednosti, prva je “kalibracija u tijeku” koja se vraća u slučaju kada je svojstvo “kalibracija “postavljeno na true a svojstvo “active” na false, druga je “Eksperiment u tijeku” koja se vraća kada je svojstvo “kalibracija” postavljeno na false a svojstvo “active” na true, treća moguća vrijednost varijable “stat” je “Uređaj je trenutno neaktivan” te se vraća ukoliko su oba spomenuta svojstva postavljena na false, dok se četvrto svojstvo “NEISPRAVNO OČITANJE” vraća ukoliko je u vrijednostima svojstava zabilježena greška. Podatke koje se prosljeđuje u komponentu “SenzorList” nema potrebe filtrirati podatke, kao ni kod daljnjeg prosljeđivanja iz komponente “SenzorList” prosljeđivanja u komponentu “Senzor” te iz komponente “Senzor” u komponentu “Graf”. Samo kod prosljeđivanja moramo obratiti pažnju koje podatke šaljemo sljedećoj komponenti.

4. Uputstva za rad.

4.1. Početna stranica i Izbornik

Nakon pokretanja, aplikacija se otvara na početnoj stranici na kojoj je vidljiv prikaz "HomeView" te naziv projekta, na lijevoj strani ekrana nalazi se izbornik koji nam omogućava navigaciju između različitih podstranica. Izbornik je vidljiv sa svih podstranica kako bi se omogućila nesmetana navigacija unutar aplikacije. Na slici 43 Možemo vidjeti prikaz naslovne stranice

Slika 43. Korisničko sučelje naslovne stranice aplikacije

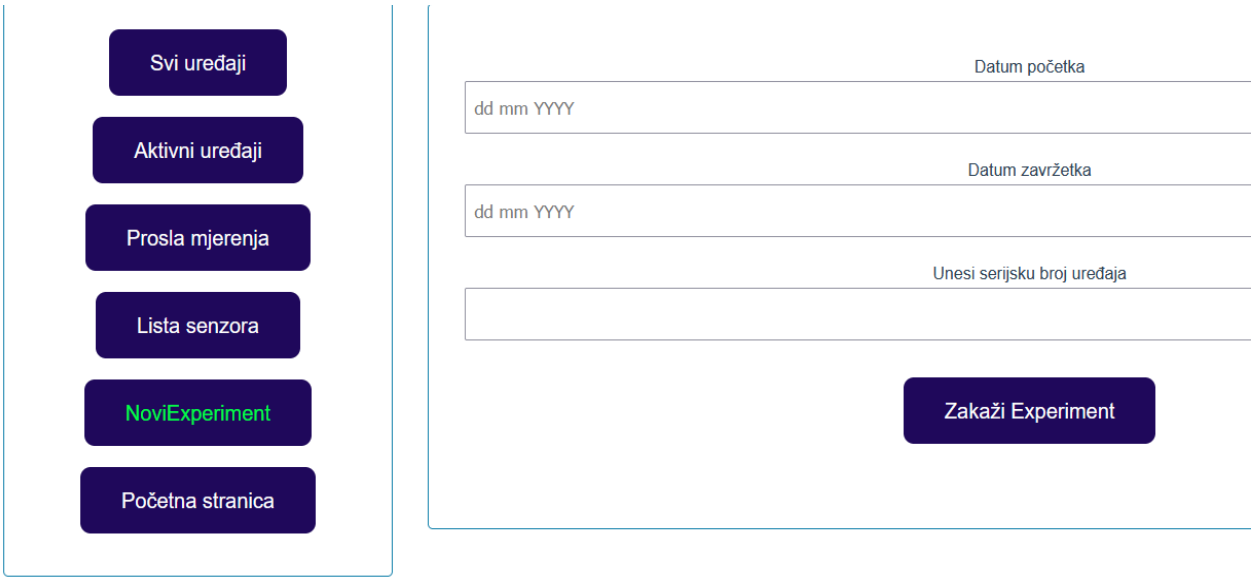


Izvor: Vlastita izrada

4.2. Pokretanje novog eksperimenta

Novi eksperiment možemo pokrenuti u pogledu "NoviEksperimentView" na kojeg prelazimo klikom na Gumb "NoviExperiment" u formu za pokretanje novog eksperimenta treba upisati datum početka eksperimenta, datum završetka eksperimenta te serijski broj uređaja koji će obavljati taj eksperiment. Datum treba pisati u formatu “DD MM YYYY” to jest dan, mjesec godina uz to da dane i mjesece koji imaju jednu znamenku zapišemo sa dvije, na primjer Za mjesec ožujak treba zapisati kao “03” nakon toga pritiskom na gumb “Zakaži Eksperiment”, ovo možemo vidjeti na slici 44.

Slika 44. Korisničko sučelje za pokretanje novog projekta

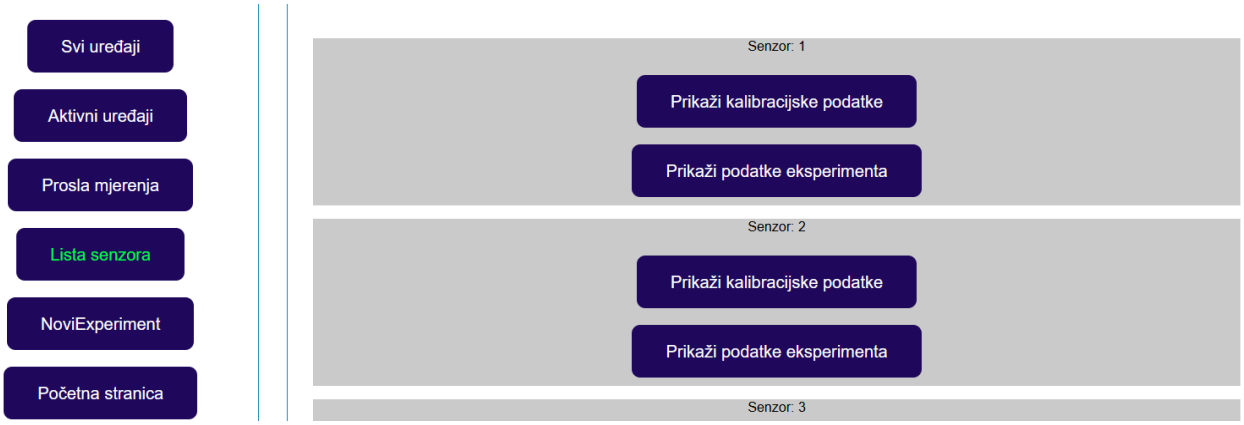


Izvor: Vlastita izrada

4.3. Prikaz liste senzora

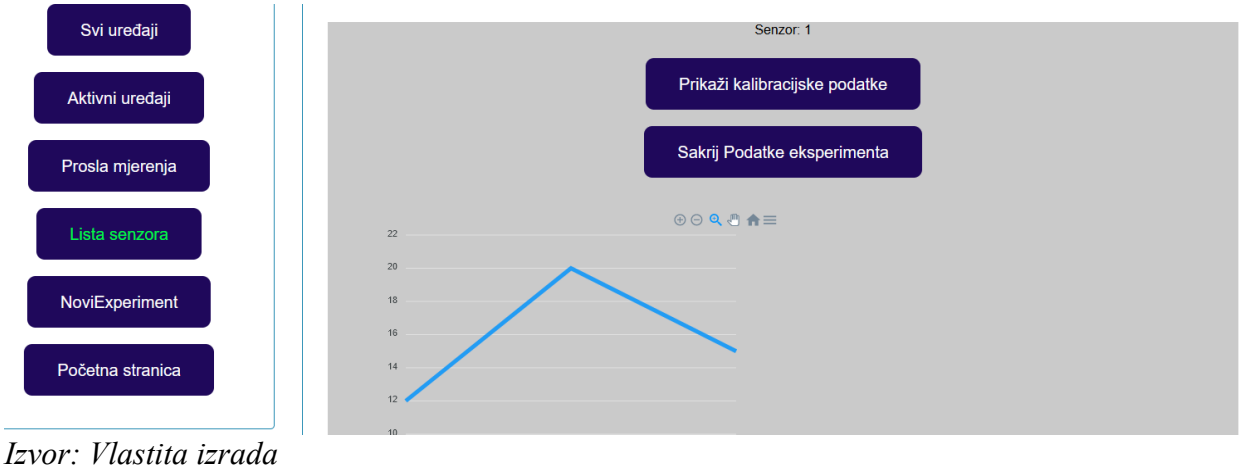
Sljedeće što možemo pregledati je lista senzora ,u kojoj za svaki senzor imamo opciju prikaza i skrivanja grafa s kalibracijskim podacima, te grafa s podacima mjerenja, ovo radimo klikom miša na gumb “prikaži kalibracijske podatke” ili “prikaži podatke eksperimenta”, koji nakon klika mijenjaju naziv u “sakrij kalibracijske podatke”, to jest “sakrij podatke eksperimenta”. Ovo možemo vidjeti na slikama 45. i 46.

Slika 45. Prikaz liste senzora sa sa sakrivenim grafovima



Izvor: Vlastita izrada

Slika 46. Prikaz liste senzora s prikazanim grafom podataka eksperimenta kod senzora 1

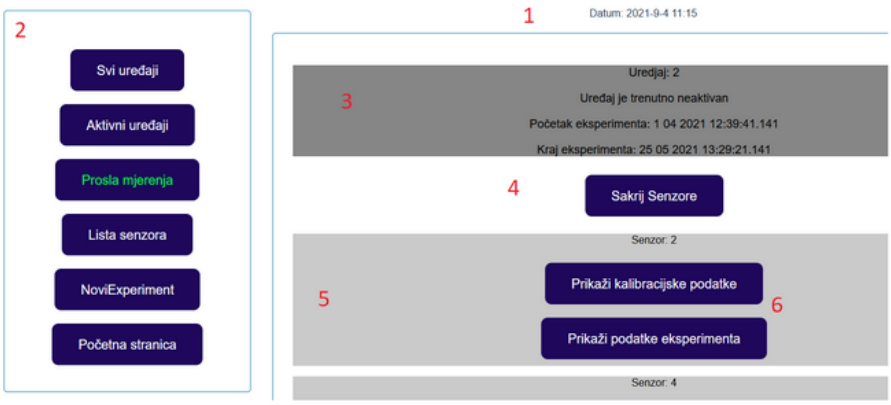


Izvor: Vlastita izrada

4.4. Prikaz Podstranica Uređaja

Gumbi “Prošla Mjerenja”, “Aktivni Uređaji” i “Svi Uređaji” koji nas vode na podstranice u kojima se prikazuju podaci o uređajima filtrirani na prije objašnjene načine. Ovdje se ispod podataka o svakom uređaju nalazi gumb “Prikaži senzore” koji omogućuje prikaz svih senzora u jednom uređaju. Na slici su 47. prikazani različiti elementi korisničkog sučelja. Na slici vidimo korisničko sučelje Podstranice koja prikazuje pošla

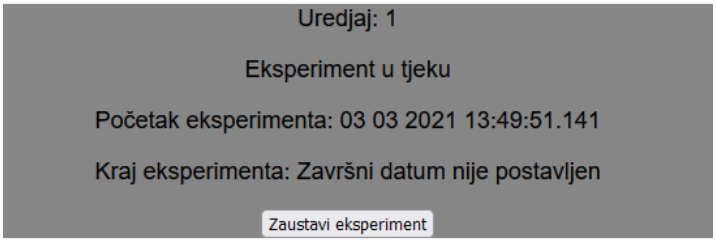
Slika 47. Prikaz Uređaja u podstranici koja prikazuje prošla mjerenja s numeracijom različitih elemenata



Izvor: Vlastita izrada

mjerenja. Brojevi na slici predstavljaju bitne elemente korisničkog sučelja. Pod brojem 1 je trenutni datum i vrijeme koji je vidljiv sa svih podstranica, kao i Izbornik označen sa brojem 2. Pod brojem 3 se nalazi prikaz podataka o uređaju, dok je gumb za prikaz podataka senzora označen brojem 4. Broj 5 označava prikaz senzora dok broj 6 označuje gumbe za prikaz kalibracijskih podataka i podataka eksperimenta jednog senzora, jedina bitna razlika između prikaza neaktivnih i aktivnih uređaja je ta da aktivni uređaji imaju uključen gumb za slanje signala za gašenje prikazanog na slici 48.

Slika 48. Prikaz podataka aktivnog uređaja s gumbom za slanje signala za gašenje na dnu prikaza

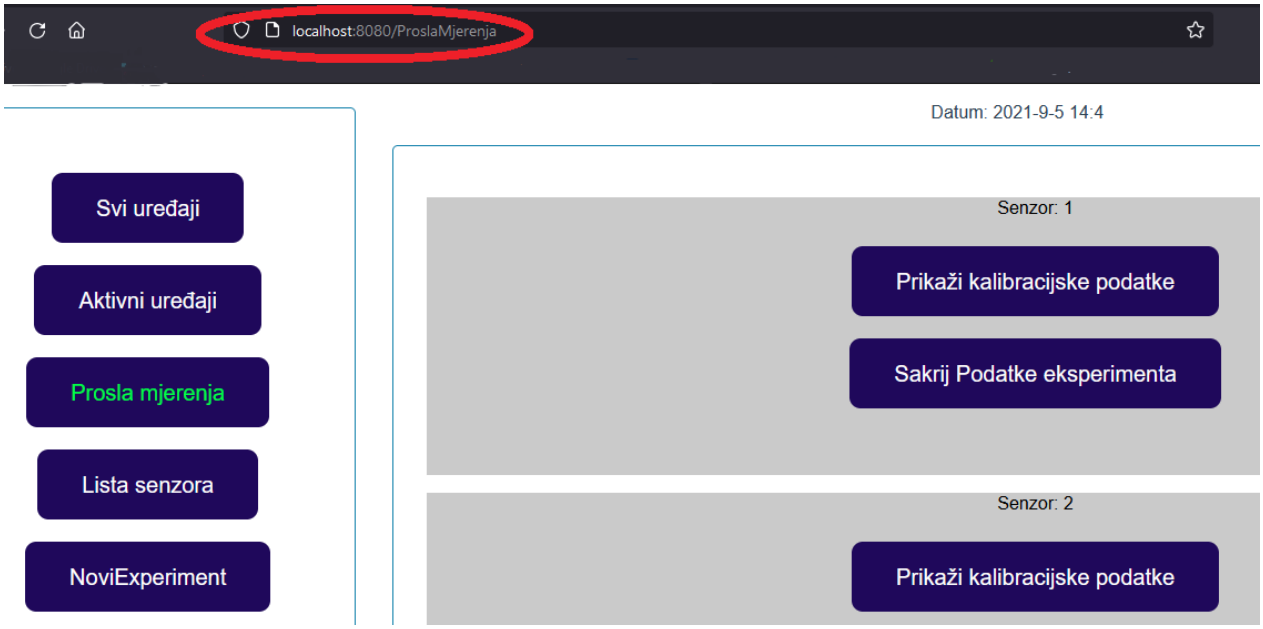


Izvor: Vlastita izrada

4.5. Trenutne greške i nedostatci aplikacije

Trenutna aplikacija, iako upotrebljiva, sadrži neke nedostatke. Najveći je taj da se kod prelaza između podstranica koje pozivaju podatke s vanjskog izvora prikaz zamrzne te stranicu treba ponovno učitati, primjer ovoga vidljiv je na slici 49. koja prikazuje zamrznut prikaz liste senzora dok u URL-u piše da se nalazimo na podstranici "ProslaMjerenja". Osim toga početnu bi se stranicu trebalo doraditi tako da izgleda primamljivije i ostavlja bolji prvi dojam te se boje pozadine i elemenata mogu bolje prilagoditi temi projekta.

Slika 49. Prikaz greške kod prelaza između dviju podstranica koje povlaće podatke s vanjskog izvora. Prikaz je neresponzivan dok se URL mjenja



Izvor: Vlastita izrada

Zaključak.

U trenutnom obliku ova aplikacija je spremna za upotrebu no u njoj i dalje ima nekih nedostataka. Kod je napisan tako da broj uređaja i senzora, također je nebitno što koji senzor mjeri, što nam omogućava da u jednom uređaju imamo senzore koji mjere temperaturu, otvorenost školjkaša, slanost vode te druge nama bitne parametre.

Sažetak

U svrhu istraživanja čistoće mora želi se pratiti otvorenost školjkaša u određenim uvjetima. Ovaj rad opisuje postupak izrade front end aplikacije sustava koji omogućava provođenje eksperimenata.

Summary

Withg the purpous of examining sea water quality this project aims to track the behavior of shellfish trough different conditions in their environmenrt. This paper describes the process of developing a frontend application of a system that would allow conduction of such experiments.

Ključne riječi

Vue, Vue3, JavaScript, Apexcharts, Vue router, školjkaši

Key words

Vue3, JavaScript, Apexcharts, Vue router,shelfish

Izvori

Web izvori

Apexcharts.js-Opensource Javascript charts for your website

https://apexcharts.com/

Vue.js

https://vuejs.org/

Vue.js

https://v3.vuejs.org/

Vue CLI

https://cli.vuejs.org/

Vue Router

https://router.vuejs.org/

npm

https://www.npmjs.com/

Codebin – Online code editor

https://www.codebin.co/

GitHub

https://github.com/

Stack Overflow – Where Developers Learn, Share & Build Careers

https://stackoverflow.com/

Vue Land

Ovo je službeni discod server programskog okruženja Vue

Vue Forum

https://forum.vuejs.org/

UML Diagram Tool| UML Diagram Online| Creately

https://app.creately.com/diagram/avGnl5PdyAH/edit

Flowcharts

https://app.diagrams.net/

Vue JS Crash Course 2021- Traversy Media

https://youtu.be/qZXt1Aom3Cs

Popis slika

Slika 1. UML dijagram prve inačice aplikacije 3
Slika 2. UML dijagram druge i treće inačice aplikacije 4
Slika 3. UML dijagram trenutne inačice aplikacije 5
Slika 4. prikaz vue koda 7
Slika 5. rezultat koda sa slike 4 7
Slika 6. Korisničko sučelje Vue CLI 8
Slika 7. Primjer grafa izrađenog u apexchartu 10
Slika 8. Paketi uvezeni U Visaul Studio Code u svrhu izrade ovog projekta 11
Slika 9. Izgled JSON datoteke prve inačice aplikacije 14
Slika 10. Izgled JSON datoteke druge inačice aplikacije *15*
Slika 11. Prikaz JSON datoteke treće inačice aplikacije 16
Slika 12. Prikaz JOSN datoteke završne inačice aplikacije 17
Slika13. Implementacija naredbe NPM run backend 18
Slika 14. Implementacija Vue routera 19
Slika15. Implementacija router-link poveznica u izbornik 19
Slika16. prvi dio koda pogleda "SviUredjajiView" 20
Slika17. Drugi dio koda pogleda "SviUredjajiView" *20*
Slika 18. Treći dio koda pogleda "SviUredjajiView" *20*
Slika19. Lista pogleda u završnoj inačici aplikacije 20
Slika 20. Komponente druge inačice aplikacije *21*
Slika 21. Komponente treće inačice aplikacije *21*
Slika 22. Komponente završne inačice aplikacije *21*
Slika 23. definiranje api kratice *22*
Slika 24. Naredba Fetch *22*
Slika 25 Prikaz nefiltriranih i neuređenih podataka *23*
Slika 26. Prikaz prosljeđivanja parametara *23*
Slika 27 Primanje uređaja proslijeđenih u komponentu *23*
Slika 28. Primjer v-for petlje i prosljeđivanja zasebnih elemenata polja *24*
Slika 29. Prikaz naredbe emit *24*
Slika 30. Primanje emitanog objekta *24*
Slika 31. Prosljeđivanje podataka na datasource *25*
Slika 32. Prikaz forme za pokretanje novih eksperimenata u grafičkom sučelju *25*

Slika 33. Prikaz koda forme za pokretanje novih eksperimenata 25
Slika 34. Prvi dio koda Apexchart grafa *26*
Slika 35. Drugi dio koda Apexchart grafa *26*
Slika 36. Graf izrađen kodom prikazanom na slikama 34 i 35 ..... *26*
Slika 37. Prikaz korisničkog sučelja prve inačice aplikacije *27*
Slika 38. Prikaz korisničkog sučelja završne inačice aplikacije sa sakrivenom listom *27*
senzoraje povlaće podatke s vanjskog izvora.
Slika 39. Prikaz korisničkog sučelja *27*
završne inačice aplikacije sa otkrivenom listom senzora
Slika 40. Petlja koja neaktivne uređaje sprema u varijablu "temp" 29
Slika 41. Kod metode provjere aktivnosti uređaja 30
Slika 42 Kod metode koja postavlja statusa uređaja u tekstualnom obliku 31
Slika 43. Korisničko sučelje naslovne stranice aplikacije 32
Slika 44. Korisničko sučelje za pokretanje novog projekta 33
Slika 45. Prikaz liste senzora sa sa sakrivenim grafovima 34
Slika 46. Prikaz liste senzora s prikazanim grafom *34*
podataka eksperimenta kod senzora 1
Slika 47. Prikaz Uređaja u podstranici koja prikazuje prošla mjerenja *35*
s numeracijom različitih elemenata
Slika 48. Prikaz podataka aktivnog uređaja s gumbom *35*
za slanje signala za gašenje na dnu prikaza
Slika 49. Prikaz greške kod prelaza između dviju podstranica *36*
koje povlače podatke s vanjskog izvora.
Prikaz je neresponzivan dok se URL mijenja