

Implementacija vizualnog sustava za praćenje dijelova 3D pisača koristeći Node-RED

Mirko, Gabrijel

Undergraduate thesis / Završni rad

2025

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

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:137:269739>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-03-10**



Repository / Repozitorij:

[Digital Repository Juraj Dobrila University of Pula](#)



Sveučilište Jurja Dobrile u Puli

Tehnički fakultet u Puli

Gabrijel Mirko

**"Implementacija vizualnog
sustava za praćenje
dijelova 3D pisača koristeći
Node-RED"**

Završni rad

Pula 2025, godina.

Sveučilište Jurja Dobrile u Puli

Tehnički fakultet u Puli

Gabrijel Mirko

"Implementacija vizualnog
sustava za praćenje dijelova
3D pisaača koristeći Node-
RED"

Završni rad

JMBAG: 0303104439 , redoviti student

Studijski smjer: Računarstvo

Znanstveno područje: Tehničke znanosti

Znanstveno polje: Računarstvo

Mentor: prof. dr. sc. Sven Maričić

Pula, veljača, 2025godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani _____, kandidat za prvostupnika
_____ovime izjavljujem
da je ovaj Završni/Diplomski rad rezultat isključivo mogega 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 Završnog 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, _____, _____ godine

IZJAVA

o korištenju autorskog djela

Ja, _____ dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni 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, _____, _____godine

Potpis

Sadržaj

1	Uvod	1
2	Pisač Prusa MK3 2s	1
3	FDM (Fused Deposition modeling) tehnologija	2
4	Što je Node-RED?	2
4.1	Što je to Node (Čvor) ?	2
4.2	Što je to Flow (Tok)?	2
5	Instalacija	3
6	Sučelje NODE-RED - a	3
7	Radno sučelje	4
7.1	Lista čvorova	4
7.2	Radna Površina	5
7.3	Debug	5
8	Objašnjenje čvorova	5
9	Jednostavan primjer NODE-RED - a	6
10	Motivacija za odabir teme	10
11	Cilj projekta	10
12	Funkcionalnosti sustava	11
13	Dodavanje predmeta	13
14	Čitanje i prikaz podataka	16
15	Brisanje predmeta	22
16	Uređivanje podataka	25
17	Testiranje sustava	29
18	Problemi i izazovi tijekom izrade	30
19	Zaključak	31
	Popis literature i drugih izvora podataka koji su upotrijebljeni u izradi završnog rada	32
	Prilozi	33
	Sažetak i ključne riječi (Abstract and keywords)	34

1 Uvod

U današnjem digitalnom, s porastom online kupovine i sve pristupačnijim uređajima i alatima, upravljanje podacima postalo je ključno za efikasno poslovanje. A pogotovo u područjima kao što su skladištenje, laboratoriji, trgovine i slično. Svakodnevno se suočavamo s velikim količinama podataka koji zahtijevaju brzo, učinkovito i precizno rukovanje. Srećom razvojem tehnologija poput baza podataka, alata za pohranu i mrežnih aplikacija čini ovaj proces lakšim, bržim i učinkovitijim. Ovaj rad istražuje razvoj sustava za upravljanje skladištem temeljen na jednostavnom i učinkovitom sustavu za pohranu podataka u JSON datoteke. Kroz ovaj projekt implementirat će se funkcionalnosti kao što su unos, ažuriranje i brisanje podataka o predmetima, te njihovo kasnije dohvaćanje i prikazivanje u grafičkom sučelju. Cilj je stvoriti sustav koji korisnicima omogućava lakši i jednostavniji rad s podacima, a istovremeno minimizira mogućnost ljudskih pogrešaka i omogućava brzu pohranu i trajno spremanje podataka. Mnoge trgovine, skladišta, laboratoriji i druga poslovanja se koriste sličnim tehnologija izrade sustava za vođenje inventara. Ovaj rad također analizira prednosti i izazove implementacije takvih sustava. Završni rad napisan je u sklopu laboratorijskih aktivnosti „EduCalucemTech“ projekta.

Hipoteza:

Razvoj sustava za upravljanje skladištem temeljenog na pohrani podataka u JSON datoteke može značajno poboljšati efikasnost, točnost i brzinu upravljanja podacima u poslovnim okruženjima kao što su trgovine, skladišta i laboratoriji, smanjujući mogućnost ljudskih pogrešaka.

Predmet istraživanja:

Razvoj sustava za upravljanje skladištem temeljenog na JSON datotekama za pohranu podataka.

Problem istraživanja:

Kako implementirati sustav za upravljanje skladištem koji omogućava učinkovito upravljanje podacima uz minimiziranje ljudskih pogrešaka i omogućava brzo i sigurno pohranjivanje podataka?

Ciljevi rada:

Razviti sustav za pohranu podataka u JSON datoteke koji omogućava unos, ažuriranje, brisanje i dohvaćanje podataka o predmetima. Osigurati jednostavno i učinkovito sučelje za korisnike. Analizirati prednosti i izazove implementacije sustava za vođenje inventara u različitim poslovnim okruženjima.

Metodologija rada:

Istraživanje teorijskih osnovâ o sustavima za upravljanje skladištem i pohranu podataka u JSON formatu. Razvoj i implementacija sustava koristeći tehnologije za pohranu podataka, mrežne aplikacije i baze podataka. Testiranje i evaluacija sustava na temelju stvarnih podataka i analiza učinkovitosti u poslovnom okruženju. Analiza prednosti i izazova implementacije sustava kroz studije slučaja i literaturu.

2 Pisač Prusa MK3 2s

Prusa MK3 2s je jedan od najpopularnijih 3D pisača trenutno na tržištu. Poznat po svojoj pouzdanosti, kvaliteti ispisa i jednostavnosti korištenja. Proizvodi ga češka tvrtka "Prusa Research", a ima široku primjenu od kućnih potrepština do profesionalnih rješenja. Ovaj model koristi FDM (Fused Deposition Modeling) tehnologiju koja radi na principu ekstruzije termoplastičnog materijala sloj po sloj. Može ispisivati razne vrste polimera: PLA, PETG, ABS, ASA, TPU itd. Prikazani pisač ima mnogo karakteristika poput:

1. Automatskog kalibriranja stola
2. Senzor filamenta¹
3. Tihi rad
4. Otvoreni kod (Open-source)
5. MMU2S (mogućnost rada s više materijala)

Tehničke karakteristike:

- Radni volumen: 250 x 210 x 210 mm
- Promjer mlaznice 0,4 mm (može se zamijeniti)
- Brzina ispisa: DO 200 mm/s (preporučeno 60-80 mm/s)
- Grijana podloga



Slika 1 Prusa 3D pisač MK3 2S

3 FDM (Fused Deposition modeling) tehnologija

FDM (Modeliranje taloživog materijala) je jedna od vrsta tehnologija 3D ispisa. Najčešće korištena, koja radi na principu istiskivanja termoplastičnog materijala. Radi se o "aditivnom proizvodnom procesu", što znači da se materijal topi i nanosi se sloj po sloj, pa se tako oblikuje prema potrebi kako bi se stvorio tro-dimenzionalni objekt.

Prvo je potrebno zagrijati mlaznicu. Kroz zagrijanu mlaznicu prolazi filament, on se topi na visokim temperaturama. Rastopljeni materijal se precizno nanosi na grijanu podlogu 3D pisača. 3D pisač radi slojeve i stvara oblik željenog 3D objekta. Materijal se hladi uz pomoć ventilatora koji se nalazi na glavi 3D pisača.

Ova tehnologija je donesla malu revoluciju, zato jer je jednostavna za korištenje i jeftina.

1. Polimer u obliku tanke niti, koji se koristi za 3D tisak

Također materijali su pristupačni jer FDM ima široki spektar mogućnosti korištenja “termo- plastika²” poput: PLA, ABS, PETG itd. Koji se ne koriste samo u industrijske svrhe nego i svakodnevnim proizvodima od plastičnih boca, vrećica, pribora, igračaka itd. Svaka plastika koja se koristi je drukčija i ima svakakvih karakteristika. Neke su fleksibilne (TPU) , neke izuzetno čvrste i otporne (ABS), do nekih svakodnevnih (PETG i PLA).

Osim toga, FDM ima jednostavnu mogućnost prilagodbe. Svaki korisnik može samostalno dizajnirati i prilagoditi dijelove bez potrebe za skupim alatima i dugotrajnim proizvodnim procesima.

4 Što je Node-RED?

”Node-RED je programski alat za spajanje hardverskih uređaja, API-eva i online usluga. Pruža uređivač koji radi preko preglednika s čime lako spajamo različite Node-ove (čvorove) koji se mogu Deployati samo s jednim klikom.”

Node-RED je programski alat za vizualno programiranje preko Internetskog preglednika. Node-RED omogućuje povezivanje hardvera, API-ja i online servisa pomoću grafičkog sučelja. Koristi se za automatiziranje procesa, obradu podataka i stvaranje aplikacija temeljem događaja. Node-RED definiramo tijekom podataka (flows) uz niz čvorova (nodes).

4.1 Što je to Node (Čvor) ?

Node (čvor) je osnovna komponenta u Node RED-u. Povezivanjem više čvorova (nodes) možemo izraditi flow (Tok), odnosno tok podataka ili procesa koji izvršava određeni zadatak.

Postoje različite vrste čvorova i svaki ima svoju određenu funkcionalnost, zadatak s kojom možemo napraviti različite stvari (Ulazne čvorove, Izlazne čvorove, Čvorovi za obrađivanje, Integracijski čvorovi, Specijalni čvorovi).

Čvorovi se međusobno povezuju s linijama kako bi se odredio točan niz radnji ili događaja.

4.2 Što je to Flow (Tok)?

U Node-RED-u flow predstavlja skup međusobno povezanih čvorova koji zajedno izvršavaju određeni zadatak. Flow se koristi za upravljanje toka podataka ili događaja kroz različite korake: od ulaza podataka, njihove obrade do izlaza. Flow-ovi se prikazuju kao dijagram gdje su čvorovi povezani linijama koje simboliziraju protok podataka.

Svaki flow definira određenu logiku ili funkcionalnost poput senzorskih podataka,

² Vrsta polimera koja se može više puta zagrijati i oblikovati

obavijesti, integracije s API-jem itd. Više flow-ova može postojati unutar istog Node-RED projekta, a svaki od njih može raditi kao izolirani proces, neovisno o ostalima.

5 Instalacija

Instalacija Node-RED je drukčija za svaki operativni sustav. Objasniti ću primjer instalacije na operativnom sustavu Windows.

Prvo je potrebno instalirati Node.js, jer je Node-red baziran na njemu. Node.js se preuzima sa službene stranice : “<https://nodejs.org>” i instalira se na računalo.

Zajedno s Node.js-om se instalira i “npm” (Node Package Manager), koji je možemo reći glavni upravitelj za Node.js.

Provjeru instalacije možemo isprobati u konzoli (Command Prompt) sa sljedećim naredbama:

```
node -v  
npm -v
```

Ako nam naredbe ne vrate nikakvu grešku znači da je alat uspješno instaliran. Za instalaciju Node-red -a u terminal se upisuje sljedeća naredba:

```
npm install-g--unsafe-permnode-red
```

Nakon par minuta izvršavanja komandi možemo provjeriti ispravnost instaliranja Node-red-a s komandom:

```
node-red
```

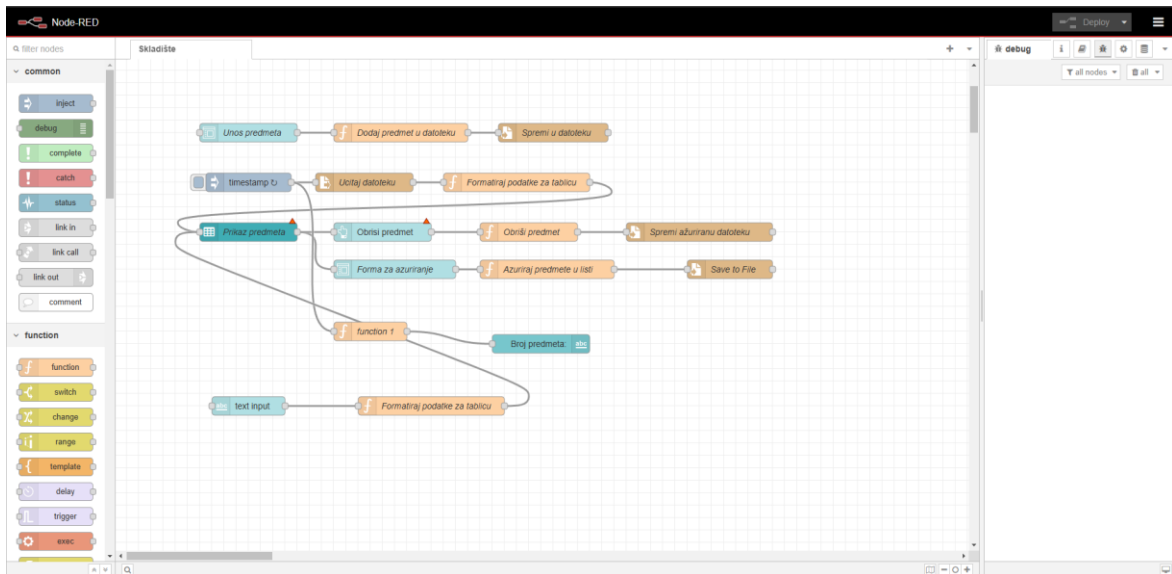
S ovom naredbom pokrećemo Node-red na prikaz na WEB- pregledniku. Za pristup sučelju unosi se adresa: “<http://localhost:1880>” kao URL za adresu preglednika. U WEB pregledniku bi nas trebao dočekati Node-red radni prostor, čiji ćemo izgled objasniti u sljedećem poglavlju.

6 Sučelje NODE-RED - a

Node-RED sučelje je jednostavno i intuitivno. Vizualni dizajn nam omogućuje brže i lakše kreiranje tokova (flows).

Za stvaranje toka (flow-a) prvo odabiremo koje ćemo čvorove koristiti. Nakon što smo odabrali čvorove trebamo ih povući iz palete (lijeva strana aplikacije) i postaviti u radni prostor. Više čvorova potrebno je povezati, povezivanje čvorova se radi tako da kliknemo na izlaz jednog čvora i povučemo na ulaz drugog čvora. Čvorovi bi se trebali sami spojiti. Ako trebamo podesiti (konfigurirati) čvor to radimo tako da kliknemo dvaput na čvor. Tada nam se otvara poseban prozor gdje možemo podesiti parametre čvora. Na kraju kad

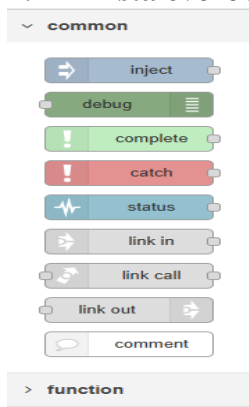
dodamo i uredimo sve čvorove kako smo željeli potrebno je Deploy-ati tok. To se radi tako da se klikne na tipku Deploy (desna strana na vrhu) i tok bi se trebao pokrenuti.



Slika 2 Izgled Sučelja

7 Radno sučelje

7.1 Lista čvorova



Slika 3 Lista čvorova

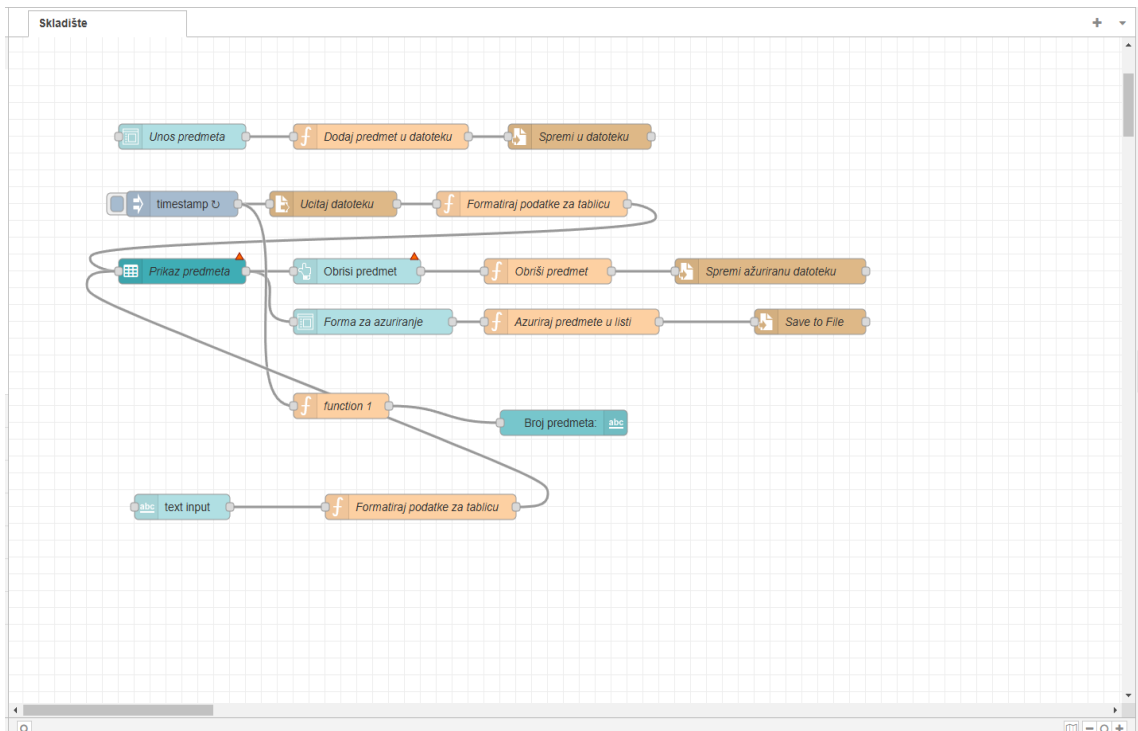
Na lijevoj strani (Node-RED -a) se nalaze svi čvorovi koje možemo dodati u naš projekt. Potrebno je samo odabrati čvor, kliknuti i povući na radno polje. Čvorovi su osnovne jedinice u NODE-RED-u. One omogućavaju obradu, prijenos i upravljanje podacima unutar aplikacije. Svaki čvor ima specifičnu funkcionalnost i povezuje se s drugim čvorovima kako bi se izgradila željena logika automatizacije. Lista čvorova lijevo se također i naziva Paleta čvorova, to je lista svih dostupnih čvorova koje se mogu koristiti u projektu. Svi čvorovi su podijeljeni u svoje različite kategorije, poput: Common, Function,

Network, Sequence itd. Uz te osnovne čvorove koji se već nalaze u NODE-RED-u, korisnici mogu instalirati dodatne čvorove preko raznih zajednica ili preko internetskih aplikacija. Ti dodatni čvorovi proširuju funkcionalnosti platforme. U sljedećem poglavlju biti će objašnjeni čvorovi iz pojedine kategorije.

7.2 Radna Površina

Radna površina se nalazi na sredini sučelja. Ovo je područje gdje kreiramo i uređujemo tokove. Povlačenjem čvorova s lijeve strane na našu radnu površinu i spajanjem njihovih priključaka linijama definiramo tijek podataka. Radnu površinu je moguće pomicati kako bismo imali veći prostor za rad s čvorovima. Isto tako ju je moguće povećati ili smanjiti pregled same radne površine. U jednom projektu možemo imati više kartica (radnih površina), kako bi bolje rasporedili rad više tokova i imali bolju

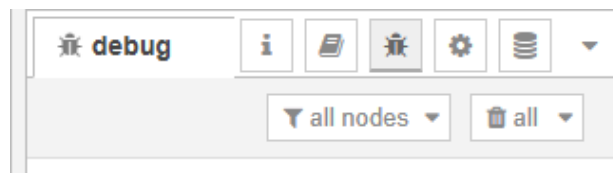
organizaciju.



Slika 4 Radna površina

7.3 Debug

Nalazi se na desnoj strani sučelja. Prikazuje poruke i podatke iz Debug čvora. On nam služi za praćenje toka podataka i pronalazak grešaka.



Slika 5 Debug

8 Objašnjenje čvorova

Common Čvorovi

U ovu kategoriju spadaju čvorovi osnovni čvorovi najčešće potrebne za rad s tokovima i čvorovi potrebni za testiranje rada toka. Dva najčešće korištena čvora su:

Inject čvor služi za pokretanje toka s nama definiranim podacima. Ima automatsko i ručno slanje podataka.

Debug čvor Prikazuje izlazne podatke/poruke u Debug Panelu (koji je objašnjen na sljedećim stranicama). Debug koristimo za provjeru ispravnosti rada našeg toka.

Funkcijski čvorovi koriste se za pisanje koda koje možemo upotrijebiti za obradu, transformaciju i filtriranje podataka.

Čvor **Function** omogućuje pisanje prilagođenog JavaScript koda za obradu podataka.

Change mijenja sadržaj poruke (msg).

Switch usmjerava tok podatak prema točno zadanim uvjetima.

Network Služe za povezivanje i komunikaciju preko mreže.

- Primanje HTTP zahtjeva i slanje odgovora
- TCP/UDP, MQTT, itd

Sequence (Nizovi) Koriste se za rad s nizovima. Oni su potrebni kad nam treba spojiti ili razdvojiti podatke (za lakše procesiranje), za sortiranje podataka, za lakše rukovanje s grupiranim ili kompliciranim podacima.

Parser (Razvrstavanje i obrada podataka) S ovim čvorovima možemo napraviti analizu i pretvorbu različitih formata podataka.

Storage Čvorovi u ovoj kategoriji omogućuju rad s datotekama, bazom podataka i pohranom.

Dashboard

Služi za stvaranje intuitivnog korisničkog sučelja (UI). Oni nam pomažu u prikazivanju podataka korisniku i omogućuju korisniku slanje povratne informacije preko sučelja. Koristan je jer nam nije potrebno prethodno znanje o HTML-u i CSS-u, a omogućuje dodavanje tipki, grafova, klizača itd.

Ova kategorija nije u standardnoj verziji Node-red, ali je ipak vrijedi spomenuti jer se jako često koristi. Dashboard je dodatak (eng. Library - Biblioteka) kojeg je potrebno posebno instalirati da bi ga se moglo koristiti, ali njegova instalacija nije uopće komplicirana.

9 Jednostavan primjer NODE-RED - a

U nastavku slijedi prikazivanje jednostavnog programa koji je napravljen u Node-red-u. Svrha ovog jednostavnog projekta je pokazati kako se projekti izrađuju u Node-red-u i kako to može biti brzo i lagano.

Tema projekta je "Štoperica." Po nazivu se može zaključiti o čemu se radi, cilj je izraditi jednostavan mjerač vremena (Kronometar) . Štoperica treba imati: tipku za početak, tipku za kraj, prikaz vremena (znamenki), tipku za ponovno pokretanje. Pošto se radi o jednostavnoj štoperici nećemo prikazivati stotinke, niti tipku i tekst za prikaz vremena jednog kruga (u slučaju potrebe za tim mogućnostima možemo ih naknadno nadodati).



Slika 6 Izgled

konačnog

grafičkog sučelja za štopericu

Kako se koristi

Tok ima tri tipke: Pokreni, zaustavi i Reset.

Klikom na gumb Pokreni, Pokreće se štoperica. Proteklo vrijeme se počinje prikazivati u stvarnom vremenu. Vrijeme se ažurira svake sekunde.

Kako bismo zaustavili štopericu i pogledali koje je konačno vrijeme pritisnemo na tipku Zaus- tavi. Nakon zaustavljanja, prikazuje se konačno vrijeme odnosno koliko je sekundi prošlo od pokretanja.

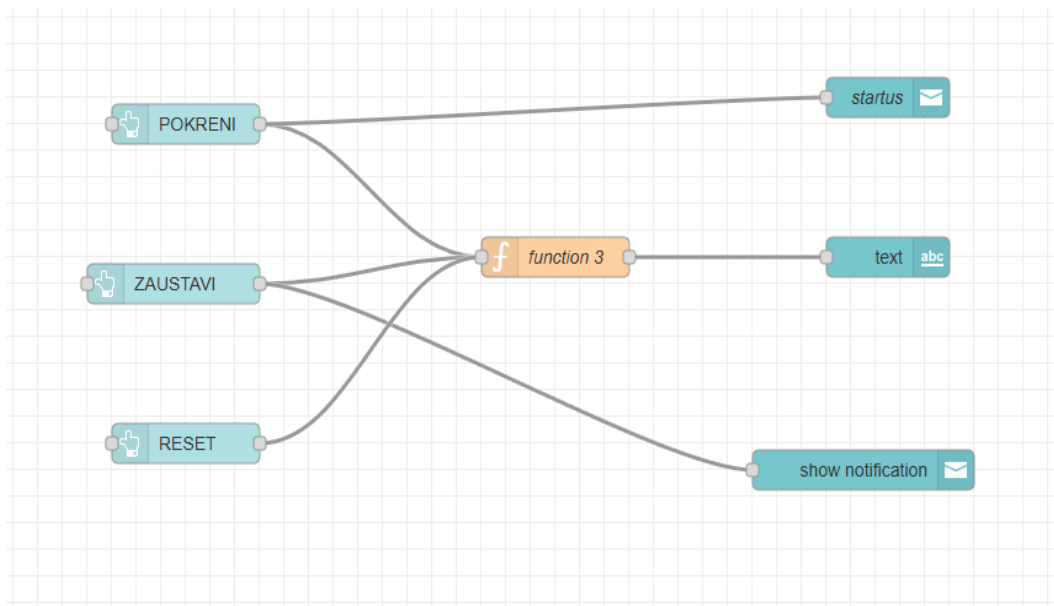
Za resetiranje štoperice kako bi se mogla ponovno pokrenuti potrebno je kliknuti na tipku Reset. Vrijeme se vraća na nulu, a štoperica je spremna za ponovno pokretanje.

Kreiran je jednostavan tok koji uključuje tri UI gumb čvora, jedan Function čvor i jedan UI Text čvor. Svaki čvor ima svoj zadatak:

UI Gumb čvorovi šalju naredbe za pokretanje (start), zaustavljanje (stop) i resetiranje štoperice.

Function čvor implementira logiku štoperice. On je “mozak” cijelog toka, odlučuje kada ce se pokrenuti, zaustaviti itd.

UI Text čvor prikazuje proteklo vrijeme na korisničkom sučelju. Također pokazuje i trenutne pritisnute naredbe (Štoperica pokrenuta, zaustavljanja, resetirana itd)



Slika 7 Izgled toka Štoperice

Programski kod

Logika štoperice implementirana je unutar Function čvora. U nastavku slijedi prikaz tog koda i objašnjenje.

```

if(msg.payload ==="start") {
  context.set("startTime",Date.now());
  context.set("interval",setInterval(()
    },1000));
  return{payload:"Štopericapokrenuta!";}
}else if(msg.payload ==="stop")
{clearInterval(context.get("interv
al"));letstartTime =
context.get("startTime");if(startT
ime) {
  letelapsed=(Date.now()-startTime)/1000;
  return{payload:`Konačno
vrijeme:${elapsed.toFixed(1)}sekundi`};
}else{
  return{payload:"Štopericanijepokrenuta!";}
}
}else if(msg.payload ==="reset")
{clearInterval(context.get("interval"));
context.set("startTime",null);
return{payload:"Štopericaresetirana."};
}
return msg;

```

Kod radi tako da na temelju ulazne poruke izvršava točno određenu radnju. Što znači da kod radi različite radnje ovisno o ulaznim naredbama, a te naredbe su: start, stop, reset. Pomoću tih naredba se štoperica pokreće, zaustavlja ili resetira.

Kada poruka (msg.payload) sadrži ključnu riječ "start", kod prepoznaje da korisnik želi pokrenuti štopericu. Zatim se izvrše niz naredbi.

```
context.set("startTime", Date.now());
```

Pohranjuje trenutni trenutak (vrijeme) kad je štoperica pokrenuta.

```
context.set("interval", setInterval(()=>{
```

setInterval omogućuje pokretanje koda unutar nje svaku sekundu. Svaki put kad se izvrši interval, štoperica će izračunati proteklo vrijeme i slati ga dalje.

```
let startTime=context.get("startTime");
```

Stvara se varijabla u koju se dohvaća i sprema prethodno pohranjeno početno vrijeme startTime, koje je postavljeno kada je štoperica pokrenuta.

```
if(startTime){
```

Ovdje se provjerava postoji li pohranjeno vrijeme. Ako je pohranjeno kod nastavlja na sljedeću naredbu:

```
let elapsed=(Date.now()-startTime)/1000;
```

Date.now() uzima trenutno vrijeme

(Date.now() - startTime) oduzima početno vrijeme od trenutnog vremena, time se dobiva proteklo vrijeme od pokretanja štoperice. Broj zatim trebamo podijeliti s 1000 kako bi se pretvorilo u sekunde.

```
node.send({payload:`Proteklo vrijeme:${elapsed.toFixed(1)}sekundi`});
```

node.send() šalje poruku (payload) prema sljedećem čvoru (UI Text)

elapsed.toFixed(1) formatira brojeve tako da se prikazuju s jednom decimalom npr. (5.6 umjesto 5.589745)

Poruka se šalje u obliku: "Proteklo vrijeme: X.X sekundi", X.X je broj sekundi. Interval se ponavlja svaku sekundu.

```
return{payload:"Štoperica pokrenuta!"};
```


Na kraju funkcije, šalje se poruka “Štoperica pokrenuta!” Poruka se prikazuje na korisničkom sučelju, kako bi se obavijestilo korisnika da se štoperica uspješno pokrenula.

Ako Funkcijski čvor dobije poruku (msg.payload) “stop” izvršava se sljedeći kod:

```
clearInterval(context.get("interval"));
```

Funkcija clearInterval() zaustavlja pokrenuto vrijeme pomoću setInterval().

context.get("interval") dohvaća vrijednost varijable koja prikazuje proteklo vrijeme i prekida je.

```
let startTime=context.get("startTime");
```

U ovu varijablu se sprema početno vrijeme koje je bilo zabilježeno kada se je štoperica pokrenula.

```
return{payload:`Konačno vrijeme:${elapsed.toFixed(1)}sekundi`};
```

Uz pomoć početnog vremena izračunava se koliko je sekundi proteklo od trenutka pokretanja štoperice do trenutka zaustavljanja. Na kraju funkcije se prikazuje poruka koja prikazuje konačno vrijeme (sa stotinkama.)

Ako nema spremljenog vremena (startTime) znaci da štoperica nije pokrenuta. U tom slučaju se korisniku na sučelju vraća poruka “Štoperica nije pokrenuta.”

Ako Funkcijski čvor dobije poruku (msg.payload) “reset” izvršava se sljedeći kod:

```
clearInterval(context.get("interval"));
```

Na isti način kao u naredbi stop, ova naredba zaustavlja interval kako bi se zaustavilo protok vremena.

```
context.set("startTime", null);
```

Naredba briše prethodno spremljeno vrijeme, tako da postavlja vrijednost startTime na vrijednost null. Štoperica se tada vraća u početno stanje u kojem se može ponovno pokrenuti.

10 Motivacija za odabir teme

Motivacija za ovu temu je nastala iz moje vlastite potrebe za organizacijom i praćenjem elektrone, elektroničkih komponenti i drugi stvari u svojoj sobi. S obzirom na to da imam mnogo elektroničkih komponenti, kablova, alata i drugih predmeta često se

dogaća da zaboravim gdje sam nešto ostavio. To stvara nelagodu i nepotreban gubitak vremena. Stoga sam odlučio stvoriti aplikaciju koja bi mi omogućila da pratim svaki predmet, lokaciju, količinu i ako treba druge relevantne informacije, kako bi organizacija postala lakša i brža. Kroz ovaj projekt želim razviti sustav koji će pomoći u sličnim situacijama, kao i poboljšati efikasnost u upravljanju skladištem predmeta.

11 Cilj projekta

Cilj projekta je stvoriti sustav za upravljanje skladištem ili laboratorija, koji omogućuje jednostavno, efikasno i brzo rukovanje podacima o skladištenim predmetima. Sustav bi trebao imati proces pohrane i obradu podataka (unos, prikaza, ažuriranja i brisanja podataka, uz mogućnost trajne pohrane.) Korisnik preko internetskog preglednika u točno određenu formu unosi podatke (predmete) u sustav, (po imenu, opisu, kategoriji, itd) koji se istodobno pohranjuje u JSON datoteku. Iz te iste datoteke se mogu pročitati svi spremljeni podaci u tablici. Klikom na pojedini redak tablice se označuje taj predmet i u drugoj formi se može urediti zabilježene podatke. Promijenjeni podaci se tada ažuriraju u datoteku i takvi se prikazuju u tablici. Ako se predmet treba izbrisati, možemo ga označiti i pritisnuti na tipku za brisanje i predmet se obriše iz tablice.

Sustav je idealan za korisnike koji žele unaprijediti učinkovitost, točnost i žele smanjiti brzinu spremanja i obrade podataka u upravljanju skladištem, laboratorijem, trgovinom itd., uz minimalne tehničke zahtjeve i troškove implementacije.

Sustav sprema sve podatke u datoteku tipa JSON koje čuvaju sve informacije o predmetima u skladištu. Svaki zapis o pojedinom predmetu sadrži:

- ID: Jedinostveni broj predmeta (mora biti jedinstven)
- Naziv: Naziv predmeta kojeg spremamo
- Opis: Detaljan opis ili karakteristika predmeta
- Količina: Broj spremljenih jedinica

JSON (JavaScript Object Notation) je format za spremanje podataka. Prednost je to što je lagan, jednostavan za pisanje i čitanje. Ima široku primjenu u programiranju, aplikacijama i web razvoju. JSON datoteke koriste nastavak (ekstenziju) .json i podaci se pohranjuju u obliku parova: “ključ-vrijednost.” Prvo se zapiše ključ, to bi bilo ime vrijednosti koje se sprema (npr: “Naziv”), a vrijednost je sadržaj koja se sprema (Npr: “3D Pisač”).

Primjer podatka u JSON datoteci:

```
{
  "ime": "FilamentABS",
  "opis": "plava",
  "kolicina": 5,
  "id": 7
}
```

12 Funkcionalnosti sustava

1. Unos novih predmeta

Korisnik preko grafičkog sučelja unosi informacije o novom predmetu kojeg želi dodati.

2. Prikaz podataka

Sustav prema intervalu od jedne sekunde učitava sadržaj JSON datoteke. Podaci se prikazuju u tablici, time se dobiva preglednost i dobiva se jednostavan pristup informacijama.

Ažuriranje podataka

Korisnik po potrebi može izmijeniti podatke o postojećim predmetim (može promijeniti Naziv, Opis, Količinu itd). Nakon što je korisnik zadovoljan s određenim promjenama pritisne tipku: “Ažuriraj” i podaci se odmah zapišu u datoteku. Time se dobiva ažurnost i dinamičnost.

3. Brisanje predmeta

Ako korisnik želi ukloniti pojedini predmet iz zapisa, može to napraviti klikom na gumb za brisanje. Nakon brisanja, podaci se uklanjaju iz tablice i datoteke.

4. Trajna pohrana podataka

Sustav zapisa informacija je dinamičan. Što znači da se svaka promjena ili unos podataka automatski sprema u datoteku i ta promjena se odmah vidi u tablici. Time se dobiva očuvanje podataka i nakon zatvaranja sustava.

Prednosti sustava

Jednostavnost - Intuitivno grafičko sučelje omogućuje brzo i lako rukovanje sustavom.

Preciznost - Proces spremanja i uređivanja podataka smanjuje mogućnost ljudske pogreške.

Sigurnost pohrane - JSON datoteka pružaju pouzdan način čuvanja podataka, koji je također i pregledan. Također sustav osigurava integritet podataka kao što je na primjer provjera točnog unosa.

Modularnost - Sustav je lako nadogradiv i prilagodiv; potrebama korisnika ako za tim ima potrebe.

13 Dodavanje predmeta

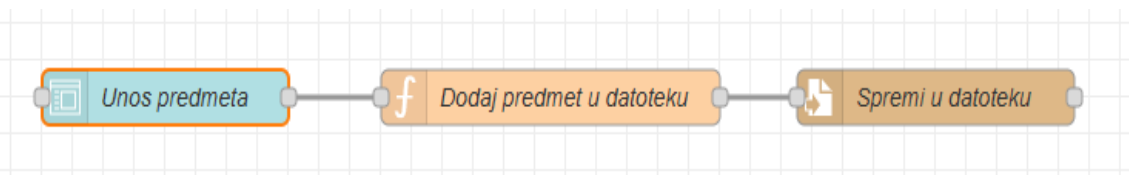
Dodavanje predmeta u sustav/datoteku započinje s Formom. U formu korisnik zapisuje potrebne informacije o predmetu kojeg unosi u sustav. (Naziv, Opis, Količina).

Forma sadrži polja koja omogućuju unos ključnih informacija o predmetu (Naziv, Opis, Količina itd.) Korisnik mora unijeti naziv predmeta, kratak opis koji opisuje distinktivne karakteristike i količinu dostupnih jedinica. Nakon što popuni sva potrebna polja, korisnik klikne na gumb za potvrdu te se podaci šalju sustavu.

Kad korisnik pošalje podatke, oni se obrađuju. Najprije se generira jedinstveni identifikator (ID) za svaki novo dodani predmet. ID osigurava da je svaki zapis jedinstven tj. da niti jedan zapis ne može biti isti i da ih možemo razlikovati. Isto tako omogućuju lako praćenje i upravljanje podacima. Sustav koristi brojač koji se mijenja ovisno o ulaznim podacima u datoteci odnosno aplikaciji. Korisnik sam ne može mijenjati ID jer može zabunom napisati krivi ili već postoji ID. Automatskim unosom se smanjuje šansa za nastajanje pogreške.

Sustav onda podatke dodaje u već zadanu strukturu podataka koja će se koristiti za pohranu podataka (JSON datoteka.) JSON format omogućuje jednostavnu i laku pohranu i kasnije dohvaćanje podataka, jer je čitljiv za računalo a i za ljude.

Dodavanje novog zapisa završava automatskim ažuriranjem tablice u korisničkom sučelju. Ovo je korisno za korisnika jer može odmah vidjeti novo nadodani predmet u sustavu, bez potrebe za ručnim osvježavanjem stranice ili ponovnim učitavanjem. Tako proces unosa novih predmeta teče brzo i efikasno, a korisnik s tim dobiva potvrdu da je njegov unos uspješno obrađen.

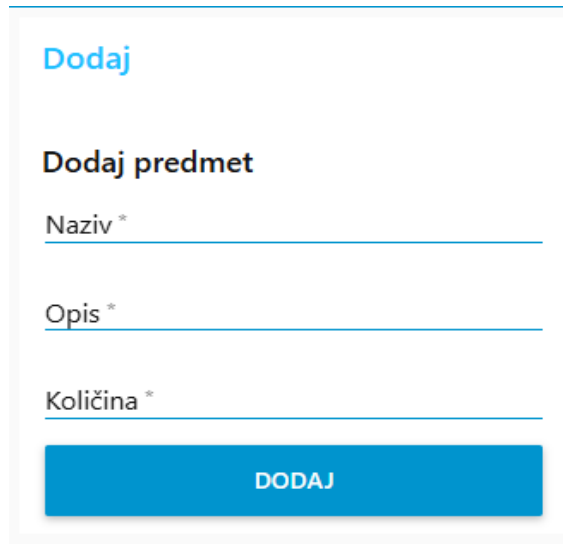


Slika 8 Dio čvorova za dodavanje novog predmeta

Objašnjenje rada toka

Prvi korak je dodati čvor “Forma” iz palete (Lijeva strana programa). Nakon što smo dodali formu trebamo urediti njezine unutarnje parametre. Trebamo dodati svaki element koji će definirati potrebne parametre/podatke o predmetima u našem skladištu. To bi bili ime ključa i ime koje će se prikazivati; u ovom slučaju nema potrebe za različitim imenima pa ćemo i ključ i vrijednost nazvati isto. Samo ćemo sadržaj napisati malim slovom kako bismo mogli razlikovati u kodu ako nam se pojavi neočekivana greška. Dodani elementi biti će: Naziv, Opis i Količina. Ako imamo potrebu dodati još

elementa možemo to lako dodati tako što pritis- nemo na tipku: “+ element” i uredimo parametre. U padajućem izborniku odaberemo koja će biti vrijednost elementa (ovisno o tipu to može biti Text,broj, znak itd.) i stavimo kvačicu da je obavezno element ispuniti s podacima (Tipka Required).



Slika 9 Forma “Dodaj predmet“

Čvor funkcija

Čvor Forma trebamo spojiti na čvor Funkcija. Ovaj funkcijski čvor nam služi da možemo podatke iz forme obraditi i kasnije poslati da se mogu spremiti. Kod u čvoru izgleda ovako:

```
let items = flow.get('items') ||
[]; let id = items.map(item =>
item.id); let newId = 1;
while(id.includes(newId)) {
    newId++;
}
msg.payload.id =
newId; items.push(msg.payload); flow
.set('items', items);
msg.payload = JSON.stringify(items, null, 2);
return msg;
```

```
let items = flow.get('items') || [];
```

flow.get('items') dohvaća podatke iz spojene forme (Naziv, Opis, Količina)

```
let id = items.map(item => item.id);
```

Dohvaćene podatke sprema u varijablu items.

Iz niza "items" vadi samo ID-jeve svih postojećih stavki i sprema ih u drugi niz "id."

"map(item => item.id)" - iterira (prolazi) kroz sve stavke i gleda samo polje "id."

```
let newId = 1;
```

Početna vrijednost "newId" (novih ID-jeva) je 1

```
while (id.includes(newId)) {  
    newId++;  
}
```

"while" petlja provjerava cijeli niz i gleda ako postoji "newId" u "id" nizu. Ako postoji, povećava "newId" dok ne pronađe prvo slobodno mjesto (broj)

```
msg.payload.id = newId;
```

Poruku postavlja na vrijednost novog ID-ja.

```
items.push(msg.payload);
```

Dodaje preko poruke novi ID u "items" niz.

```
flow.set('items', items);
```

Sprema ažurirani popis “items“ u tok (flow), i tako ostaje dostupan za buduće poruke

```
msg.payload=JSON.stringify(items, null, 2);
```

Pretvara niz items u JSON string i to s formatiranim redovima za bolju čitljivost. (null, 2 dodaje uvlačenje)

```
return msg;
```

Vraća poruku koja sadrži JSON string s novim podacima o unosu predmeta.

Čvor Spremi u datoteku

Nakon što su se podaci obradili u funkcijskom čvoru podacima se onda spremaju u datoteku uz pomoć čvora za spremanje podataka u datoteke. Prvo trebamo odrediti putanju u kojoj se nalazi JSON datoteka za spremanje podataka (Kako bi sustav mogao znati gdje će spremati podatke).

14 Čitanje i prikaz podataka

Čitanje i prikaz podataka temelji se na JSON datoteci koja sadrži informacije o predmetima u skladištu. Kad se sustav pokrene ili korisnik izvrši neku radnju iz koje se čitaju podaci, sadržaj pohranjen u JSON datoteci se obrađuje i pretvara i s njime se prikazuje u tablici.

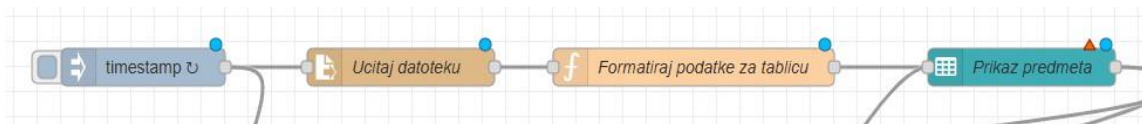
Izlazni podaci se pripremaju tako da se iz JSON datoteke očitaju svi predmeti, s njihovim vri- jednostima poput ID-ja, naziva, opisa i količine.

Svaka stavka iz JSON datoteke predstavlja jedan redak u tablici, gdje su pojedinačna svojstva predmeta prikazana u zasebnim svojstvima. Na primjer:

- Stupac “ID” prikazuje jedinstveni identifikator svakog predmeta, čime se osigurava lako prepoznavanje i razlikovanje predmeta
- Stupac “Naziv” prikazuje ime predmeta kojeg je korisnik definirao

Prilikom prikaza, podaci se formatiraju kako bi bili pregledni i lako čitljivi. Svaka izmjena podataka, poput dodavanja, ažuriranja ili brisanja, odmah se odražava u tablici jer sustav ažurira JSON datoteku i ponovno generira prikaz tablice.

Tablica nije samo alat za pregled podataka već i interaktivni element sustava. Korisnici mogu klikom na određeni redak odabrati predmet za uređivanje ili brisanje. Na taj način, tablica postaje središnji element korisničkog sučelka koji omogućava jednostavan pregled i upravl- janje podacima o skladištu.



Slika 10 Dio čvorova za čitanje i prikaz podataka

Sve započinje s “Inject” čvorom (čvor za umetanje) koji je pokretač cijelokupnog dijela toka koji se odnosi na učitavanje i prikaz podataka. On pošalje signal (poruku) daljnim čvorovima da se može započeti radnja, a to je prvo čitanje JSON datoteke. On je važan čvor jer bez njega se podaci ne bi mogli ažurirati (niti u tablici za prikaz niti u datoteci.) Umjesto toga bismo morali svaki put pritisnuti tipku za osvježavanje cijele stranice (tipka koja se najčešće nalazi gore lijevo u internetskom pregledniku). Što je ovdje bespotreban, naporan i dugotrajan proces, i još ako unosimo puno podataka i želimo vidjeti kako se oni odmah ubacuju u tablicu bismo ga morali pritiskati jako često. Zato imamo Injection čvor, koji je podešen da se osvježava svake sekunde. Ako imamo potrebe za kraćim intervalom, čvor možemo lako podesiti da brže pokreće radnje za čitanje podataka ili ako želimo smanjiti opterećenje na sustav (što u sadašnje vrijeme nije toliko veliki problem) možemo ga povećati.

Slijedi nam čvor za čitanje datoteke. Nakon što Injection čvor pokrene cijeloukupni proces aktivira se čvor za čitanje koji pročita JSON datoteku spremljenu na naše računalo. Nakon čitanja datoteke čvor šalje pročitane podatke daljnim čvorovima. Što je u ovom slučaju Funkcijski čvor.

Funkcijski čvor “Formatiraj podatke u tablicu” nam kao što samo ime kaže služi da podatke iz datoteke obradi i pripremi za prikazivanje na tablici. On iz JSON datoteke razdvaja svaki pojedini predmet i još svaki pojedini predmet još razdvoji na vrijednosti tog predmeta. Takve pojedino razdvojene vrijednosti od predmeta pojedinačno šalje u tablicu gdje se oni onda uredno prikazuju za pogled korisnika.

Prikaz

Broj predmeta: **4**

I...	Naziv	Opis	K...
1	Kliješta	za rezanje žica (Sj...	2
2	Strugalica	plastična	1
3	Filament	ABS	8
4	Filament	PLA	15

Slika 11 Tablica u kojoj se prikazuju svi predmeti u sustavu i njihovi detalji

Kod koji to radi izgleda ovako:

```
let items=JSON.parse(msg.payload||'[]');

msg.payload=items.map(item=>{

    return{id: item.id, ime: item.ime, opis: item.opis, kolicina:
item.kolicina };

});

return msg;
```

```
let items=JSON.parse(msg.payload||'[]');
```

msg.payload - je poruka koja sadrži ulazne podatke u obliku JSON formata (Naziv, opis itd)

```
let items=JSON.parse(msg.payload||'[]');
```

JSON.parse() - pretvara JSON niz u niz objekata

```
msg.payload=items.map(item=>{
```

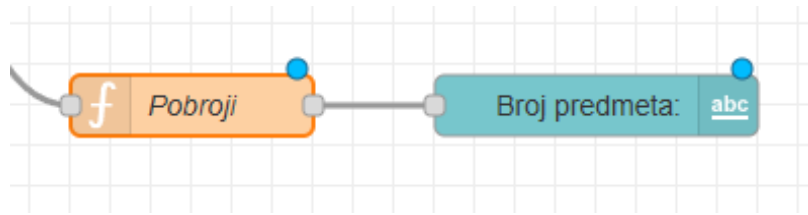
items.map() - prolazi kroz svaki objekt u nizu items.

```
return{id: item.id, ime: item.ime, opis: item.opis, kolicina: item.kolicina
};
```

za svaki predmet (item) vraća se novi objekt s željenim svojstvima

```
return msg;
```

Vraća poruku koja se šalje dalje kroz čvorove.



Slika 12 Čvorovi zaslužni za prikaz ukupnog broja predmeta u sustavu
 Također osim prikaza svih predmeta u sustavu, tok ima dva čvora koji nam služe da broje uku- pan broj predmeta u sustavu. kod je dinamičan pa se broj predmeta mijenja ovisno o broju pred- meta u sustavu (Npr. ako je u sustavu “5” predmeta i ako izbrišemo jedan predmet, prikazivat će “4”. A ako dodamo dva predmeta prikazivati će “6” predmeta u sustavu).

Brojanje predmeta započinje s funkcijom “Pobroji.” Funkcija je preko ulaza spojena na “Injection“ čvor. Injection čvor osvježava prikaz i pokreće postupak brojanja. Kod u funkciji izgleda ovako:

```
let items = flow.get('items') || [];
msg.payload = items.length;
return msg;
```

Slično kao prethodni kod, dohvaća niz “items“ iz flow memorije. Ako niz ne postoji,

```
let items = flow.get('items') || [];
```

postavlja ga na prazan niz []. (kako bi spriječio greške)

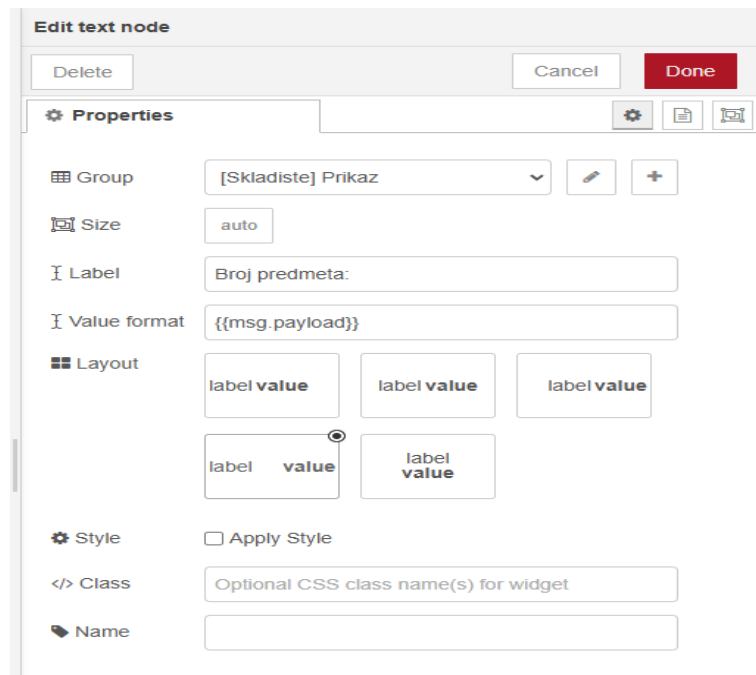
```
msg.payload=items.length;
```

“items.length” vraća broj elemenata u “items“ nizu.

Sprema taj broj u poruku (msg.payload)

```
return msg;
```

Vraća poruku s vrijednosti predmeta u nizu



Slika 13 Uređivanje UI teksta za prikaz predmeta

Na posljetku imamo UI čvor “Text“, koji na korisničkom sučelju prikazuje tekst: “Broj pred- meta:“ i prikazuje broj odnosno trenutnu vrijednost predmeta u tablici. Čvor uzima vrijednost (parametar) poruku (msg.payload) koju dobiva od prethodno povezanog čvora “Funkcijski kod.”



Slika 14 Traka za pretraživanje

Osim teksta za prikaz brojeva, na korisničkom sučelju se još nalazi traka za pretraživanje. Ona nam služi da “izolira“ (filtrira) sve nepotrebne predmete koji nam trenutno ne trebaju i da prikaže samo one koje korisnik želi, a to radi tako da korisnik u traku upiše naziv predmeta kojeg želi vidjeti količinu. Sustav tada pretražuje cijeli niz predmeta i prikazuje u tablici samo one s sličnim ili istim imenom. Kod koji to radi izgleda ovako:

```

let items=flow.get('items')||[];
let searchQuery=msg.payload.toLowerCase();

let filteredItems = items.filter(item
=>item.naziv.toLowerCase().includes(searchQuery));

msg.payload=filteredItems.map(item=>{
    return{id: item.id,naziv: item.naziv,opis: item.opis,kolicina:
item.kolicina };
});
returnmsg;

letitems=flow.get('items')||[];

```

Dohvaća popis “items“ iz flow memorije. Ako ne postoji stvara prazan niz

```
letsearchQuery=msg.payload.toLowerCase();
```

Uzima vrijednost poruke (msg.payload) koja sadrži upit za pretragu. Onda se poruka pretvara u mala slova (“toLowerCase“) kako bi pretraga bila neovisna o velikim/malim slovima.

```
letfilteredItems = items.filter(item
=>item.naziv.toLowerCase().includes(searchQuery));
```

Metoda “filter“ prolazi kroz sve predmet u nizu “items“ i vraća samo one koje imaju isti naziv kao varijabla “searchQuery.“ Svi predmeti u nizu se pretvaraju u mala slova, kako bi pretraga bila brža i neovisna o veliko/malim slovima

```
msg.payload=filteredItems.map(item=>{
    return{id: item.id,naziv: item.naziv,opis: item.opis,kolicina:
item.kolicina };
});
```

Metoda “map()“ formatira rezultate pretrage tako da vraća sva polja s podacima (id, naziv, opis, kolicina)

```
return msg;
```

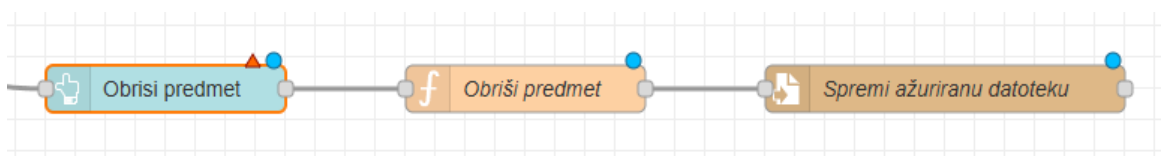
Vraća poruku s dobivenim rezultatima

15 Brisanje predmeta

Za brisanje predmeta, najprije trebamo odabrati željeni predmet koji želimo izbrisati. To se napravi tako da se dva puta klikne u tablicu na željeni redak gdje se nalazi traženi predmet. Kako bismo znali ako se predmet dobro označio, forma za uređivanje bi se trebala ispuniti s istim podacima o predmetu. Klikom na tipku “Izbrisi” predmet se briše iz tablice, nakon jedne sekunde (čim se tablica ažurira) predmet bi trebao “nestati” iz tablice.

Logički niz čvorova za brisanje predmeta je nešto jednostavniji od niza čvorova za ažuriranje predmeta. On se sastoji od tri čvora a to su redom:

Gumb “Obrisi predmet” - on ima sličnu zadaću kao “Injection čvor” a to je pokretanje radnji (za brisanje predmeta), samo što ovdje korisnik bira kada želi da se radnja pokrene. Gumb je na ulazu spojen s Tablicom koja nam prikazuje sve predmeta u našoj bazi. Bitna a skoro neprimjetna stvar je ta da u tablu za uređivanje čvora Tablice moramo imati uključeno: “Send data on click” bez toga ne bismo imali izlazni čvor na čvoru Tablice radi kojeg ne bismo mogli kliknuti na predmet tablice kojeg želimo obrisati. Radi ove tipke kada kliknemo na predmet u tablici se pošalje sve vrijednosti iz tog redka, no gumb “Obrisi predmet” gleda samo vrijednost ID-a koju kada njega pritisnemo šalje dalje poruku na izlaz čvora. Gumb je spojen s Funkcijskim čvorom gdje se pregledava i provjerava ako postoji koji je taj kliknuti ID i gleda se koji podaci su snjim povezani. Funkcija zatim obriše cijeli objekt (ID, Naziv, Opis, Količinu). Nakon brisanja cijelog redka to se sve ažurira u datoteku i odmah sprema uz pomoć čvora za spremanja u datoteku.



Slika 15 Niz čvorova za brisanje predmeta

```
let items = flow.get('items') || [];  
  
let idToDelete = msg.payload.id;  
  
items = items.filter(item => item.id !== idToDelete);  
  
flow.set('items', items);  
  
msg.payload = JSON.stringify(items, null, 2);  
  
return msg;
```

Dohvaća se varijabla items iz toka. Ako varijabla ne postoji inicijalizira se kao prazan niz.

```
let items=flow.get('items')||[];
```

```
let idToDelete=msg.payload.id;
```

Ovdje se uzima ID iz poruke (msg.payload) koji predstavlja index (ID) predmeta kojeg

```
items=items.filter(item=>item.id!==idToDelete);
```

želimo obrisati.

Ovdje se koristi metoda filter za pronalazak (filtriranje) svih predmeta iz niza items koji imaju različite ID-jeve od varijable idToDelete. To osigurava da svi predmeti koji nemaju traženi ID ostati u nizu (nepromijenjeni), a onaj koji ima taj traženi ID biti će uklonjen.

Ova linija koda sprema promijenjeni niz items u tok, kako bi se promijenjeni (ažurirani)

```
flow.set('items', items);
```

podaci mogli koristiti dalje u toku (funkcijama ili čvorovima).

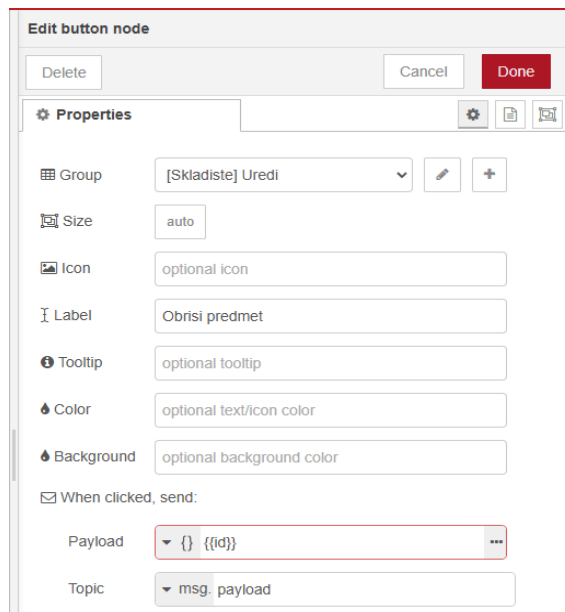
Niz items se ovdje pretvar u JSON string pomoću metode “stringify(.” JSON string se

```
msg.payload=JSON.stringify(items, null, 2);
```

for- matira s 2 razmaka radi bolje preglednosti tijekom čitanja.

Vraća poruku koja sadrži promijenjen niz items kao JSON string.

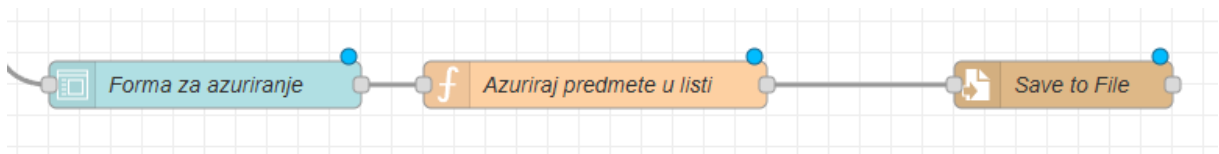
```
return msg;
```



Slika 16 Uređivanje konfi tipke za brisanje predmeta

16 Uređivanje podataka

Skup čvorova za izmjenu podataka je sličan kao i za brisanje predmeta iz tablice. Samo što ovdje umjesto tipke za brisanje imamo formu za uređivanje i funkcija je malo drugačija.



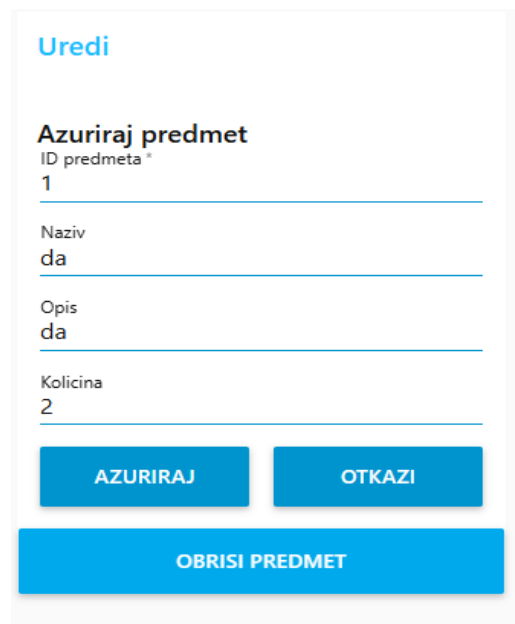
Slika 17 Niz čvorova za ažuriranje predmeta

Kako bi se moglo urediti već postoji predmet u bazi podataka, korisnik prvo mora odabrati predmet koji želi izmjeniti. Izmjena se temelji na njegovom ID-u zato jer on je jedinstven za svaki zapis (predmet). To omogućuje da sustav pronađe točno određeni predmet u našoj postojećoj bazi podataka.

Nakon odabira predmeta iz tablice i identifikacije predmeta, tablica u kojoj se nalaze svi predmeti i u kojoj je korisnik odabrao željeni predmet šalje podatke predmeta preko čvora dalje drugom čvoru, a taj čvor je forma za uređivanje predmeta. Forma se automatski ispuni s podacima iz prethodno odabranog čvora. Nakon toga korisnik može u formi uređivati podatke kako želi; Može mijenjati samo jedan znak ili cijelu riječ ako želi, a može i mijenjati sve: (osim ID-a) Naziv, Opis, Količina itd. Uneseni podaci zamjenjuju stare vrijednosti u sustavu, dok nepromijenjene vrijednosti ostaju iste.

Sustav nakon unosa novih podataka ažurira tablicu predmeta. Kako bi se osigurala trajnost promjena, ažurirana lista predmeta automatski se mora spremiti u JSON datoteku. Koji radi čak i nakon ponovno pokretanja sustava.

Nakon uspješno izvršenih izmjena, korisniku se prikazuje ažurirana tablica podataka koja sadrži nove vrijednosti. Time se dobiva transparentnost i potvrda da je uređivanje uspješno izvršeno.



Uredi

Azuriraj predmet

ID predmeta *
1

Naziv
da

Opis
da

Kolicina
2

AZURIRAJ OTKAZI

OBRISI PREDMET

Slika 18 Forma za uređivanje predmeta
Funkcijski čvor “Ažuriraj predmete u listi” koji sadrži kod za ažuriranje predmeta u listi

```
let items=flow.get('items')||[];  
  
if(!Array.isArray(items)){  
    items=[];  
    flow.set('items',items);  
}  
  
let updatedItem=msg.payload;
```

izgleda ovako:

```
if (typeof updatedItem !== 'object' || Array.isArray(updatedItem)) {  
  
    node.error("Invalid input: payload must be a JSON object", msg);  
  
    return null;  
  
}  
  
if (!updatedItem.id) {  
  
    node.error("ID is missing in payload", msg);  
  
    return null;  
  
}  
  
let index = items.findIndex(item => item.id === updatedItem.id);  
  
if (index !== -1) {  
  
    items[index] = { ...items[index], ...updatedItem };  
  
    flow.set('items', items);  
  
    msg.payload = items;  
  
    return msg;  
  
} else {  
  
    node.error(`Item with ID ${updatedItem.id} not found`, msg);  
  
    return null;  
  
}
```

```
let items=flow.get('items')||[];
```

Dohvaća varijablu items iz toka. Ako varijabla ne postoji tj. dohvati se vrijednost “null” onda se inicijalizira prazna lista.

```
if(!Array.isArray(items)){  
    items=[];  
    flow.set('items',items);  
}
```

Ovaj provjerava ako je items niz (Array), ako nije tada se postavlja vrijednost items na prazan niz, a nova vrijednost se sprema u kontekst toka.

```
letupdatedItem=msg.payload;
```

Ova linija postavlja varijablu “updatedItem” na vrijednost poruke (msg.payload), koja označava podatke koji dolaze u poruci.

```
if(typeofupdatedItem!=='object'||Array.isArray(updatedItem)){  
    node.error("Invalidinput:payloadmustbeaJSONobject",msg);  
    returnnull;}  
}
```

Ovdje se provjerava je li “updatedItem” objekt. Ako nije objekt javlja se greska i zaustavlja se funkcija.

```
if(!updatedItem.id){  
    node.error("IDissinginpayload",msg);  
    returnnull;}  
}
```

Ova linija provjerava ako objekt “updatedItem” sadrži polje id. Ako nema to polje javljase greska i funkcija se zaustavlja.

```
let index=items.findIndex(item=>item.id===updatedItem.id);
```

Ovdje se koristi metoda findIndex za traženje indeksa (ID-ja) objekta u items koji ima isti id kao i updatedItem. Ako taj objekt postoji metoda vraća njegov indeks (odnosno ID).

```

if(index!==-1){
    items[index]={...items[index],...updatedItem};
    flow.set('items',items);
    msg.payload=items;
    returnmsg;
}else{
    node.error(`ItemwithID${updatedItem.id}notfound`,msg);
    return null;
}

```

U ovoj grani ako index različit od -1, označava da je pronađen element u items s istim ID-jem. Tada se koristi sintaksa za širenje objekta “...items[index], ...updatedItem” kako bi se spojile već postojeće vrijednosti tog elementa s novim vrijednostima iz updatedItem. Niz items se ažurira i sprema natrag u kontekst toka. Onda se postavlja poruka na ažurirani niz items i ta poruka se vraća.

A ako element nije pronađen (index==1), javlja se greška i funkcija se zaustavlja.

17 Testiranje sustava

Testiranje sustava uključivalo je ispitivanje svih ključnih funkcionalnosti kako bi se osigurala njegova ispravnost, stabilnost, dinamičnost i pouzdanost.

Testiranje se je provelo kroz nekoliko scenarija koji pokrivaju osnovne operacije poput unosa novih predmeta, ažuriranja podataka o već postojećim predmetima, brisanja predmeta te prikaza svih trenutnih predmeta u skladištu.

Tijekom testiranja sustav je uspješno:

- Generirao jedinstvene ID-jeve za svaki predmet (i kasnije popunjavao prazna mjesta bez ID-jeva nakon brisanja predmeta)
- Omogućio dinamično osvježavanje prikaza podataka u tablici nakon svake izmjene (dodavanje, brisanje i ažuriranje)
- Prikazao ukupan broj predmeta u sustavu
- Automatski popunio formu s podacima iz tablice (sustav prepoznaje odabir redka gdje se nalazi pojedini predmet i automatski ažurira formu s točnim podacima)

Posebno se je trebalo obratiti pozornost na rubne slučajeve, kao što su: pokušaj dodavanja predmeta s nepotpunim ili netočnim podacima, pokušaj dodavanja simbola ili dijakritičkih znakova (poput +,@,(,š,đ itd), pokušaj dodavanja ne-numeričke vrijednosti u stavku količina.

Testirano je i spremanje i učitavanje podataka iz vanjske JSON datoteke kako bi se vidjelo ako sustav može trajno očuvati podatke koji se mogu vidjeti i nakon ponovno pokretanja sustava.

Rezultati testiranja pokazali su da sustav radi prema očekivanjima, uz stabilno, pouzdano i dinamično upravlja podacima bez greske ili zastajanja.

18 Problemi i izazovi tijekom izrade

Tijekom razvoja sustava za upravljanje skladištem u NODE-RED-u, bilo je potrebno savladati nekoliko izazova kako bi se mogao izraditi pouzdan i funkcionalan sustav.

Prvi korak bio je instalacija i razumijevanje rada NODE-RED-a. Proces instalacije nije zahtijevao mnogo vremena, a osnove korištenja brzo su savladane. Nakon razdoblja (od par sati) eksperimentiranja s funkcijama i čvorovima, stečeno je dovoljno znanja za početak izrade sustava.

Sustav je razvijan postupno, počevši od osnovnih funkcija poput dodavanja, brisanja i prikaza podataka. Ove funkcionalnosti implementirane su bez većih problema.

Najveći izazov pojavio se pri implementaciji funkcije ažuriranja podataka. Prvi pokušaji nisu bili uspješni jer se cijeli predmet duplicirao umjesto da se promijeni samo određena informacija. Kako bi se riješio ovaj problem implementirana je identifikacija predmeta pomoću ID-ja. Dodana je logika koja prepoznaje predmet prema njegovom ID broju, omogućujući ažuriranje samo onih polja koja su izmijenjena, bez stvaranja duplikata ili neispravnog izmjenjivanja.

Dodatne poteškoće nastale su nakon implementacije pokretanja funkcije “brisanja predmeta”, jer je poremetila način ažuriranja podataka. Problem je riješen tako da se umjesto forme za brisanje dodaje tipka koja je odvojena od funkcije za ažuriranje. Nakon ove prilagodbe, sustav je radio ispravno.

Ostali dijelovi sustava nisu zahtijevali značajne izmjene. Kada bi se pojavila potreba za novim funkcionalnostima, rješenja su se brzo pronalazila kroz analizu dokumentacije i primjere.

19 Zaključak

U ovom radu razvijen je sustav za upravljanje skladištem temeljen na NODE-RED-u, s ciljem jednostavnog i učinkovitog vođenja evidencije skladišnih predmeta. Implementirane su ključne funkcionalnosti, uključujući dodavanje, brisanje, uređivanje i prikaz podataka, a svi podaci pohranjuju se u JSON datoteku, što omogućava trajnu pohranu i jednostavnu obradu. Testiranje sustava potvrdilo je njegovu funkcionalnost, pouzdanost i efikasnost u radu, dok dinamičko osvježavanje podataka, automatska dodjela jedinstvenih ID-jeva i intuitivno korisničko sučelje značajno olakšavaju upravljanje podacima.

Iako su tijekom razvoja sustava postojali određeni izazovi, poput optimizacije toka podataka i organizacije korisničkog sučelja, oni su uspješno riješeni kroz testiranje i daljnja poboljšanja sustava. Razvijeni sustav može biti vrlo koristan za manja skladišta ili osobnu organizaciju, no postoji i potencijal za daljnji razvoj i prilagodbu sustava većim poslovnim okruženjima.

Zaključno, razvoj ovog sustava pokazuje kako jednostavna, ali učinkovita rješenja mogu značajno unaprijediti organizaciju skladišnih procesa, smanjiti gubitak vremena i povećati

efikasnost upravljanja resursima. Ovaj sustav pruža osnovu za daljnje istraživanje i implementaciju sličnih rješenja u različitim poslovnim okruženjima.

Popis literature i drugih izvora podataka koji su upotrijebljeni u izradi završnog rada

Blog Prusa3D. (n.d.). *Original Prusa i3 MK2 izdanje*. Preuzeto 25. siječnja 2025. s https://blog.prusa3d.com/original-prusa-i3-mk2-release_4332/

Maker Hacks. (n.d.). *Recenzija Prusa MK3S*. Preuzeto 27. siječnja 2025. s <https://www.makerhacks.com/prusa-mk3s-review/>

Wikipedia suradnici. (n.d.). *Prusa i3*. Wikipedia, slobodna enciklopedija. Preuzeto 29. siječnja 2025. s https://en.wikipedia.org/wiki/Prusa_i3

Node-RED. (n.d.). *Službena web stranica Node-RED-a*. Preuzeto 17. prosinca 2025. s <https://nodered.org/>

Node-RED Discourse. (n.d.). *Forum zajednice Node-RED-a*. Preuzeto 8. veljače 2025. s <https://discourse.nodered.org/>

Node-RED. (n.d.). *Početak rada na Windowsu*. Preuzeto 18. prosinca 2025. s <https://nodered.org/docs/getting-started/windows>

npm. (n.d.). *Node-RED paket na npm-u*. Preuzeto 20. prosinca 2025. s <https://www.npmjs.com/package/node-red>

InfluxData. (n.d.). *Vodič za instalaciju Node-RED-a*. Preuzeto 28. studeni 2025. s <https://www.influxdata.com/blog/guide-installing-node-red/>

GitHub. (n.d.). *Node-RED repozitorij*. Preuzeto 8. veljače 2025. s <https://github.com/node-red/node-red>

Reddit. (n.d.). *Node-RED zajednica na Redditu*. Preuzeto 26. studeni 2025. s <https://www.reddit.com/r/nodered/>

YouTube. (n.d.). *Node-RED YouTube kanal*. Preuzeto 14. siječnja 2025. s <https://www.youtube.com/@Node-RED>

Popis slika

Slika 1: Prusa 3D pisac MK3 2S	2
Slika 2: Izgled Sučelja	4
Slika 3: Lista čvorova	4
Slika 4: Radna površina	5
Slika 5: Debug.....	5
Slika 6: Izgled konačnog grafičkog sučelja za štopericu	6
Slika 7: Izgled toka Štoperice.....	7
Slika 8: Dio čvorova za dodavanje novog predmeta	12
Slika 9: Forma “Dodaj predmet“	13
Slika 10: Dio čvorova za čitanje i prikaz podataka.....	15
Slika 11: Tablica u kojoj se prikazuju svi predmeti u sustavu i njihovi detalji	16
Slika 12: Čvorovi zaslužni za prikaz ukupnog broja predmeta u sustavu.....	18
Slika 13: Uređivanje UI teksta za prikaz predmeta.....	19
Slika 14: Traka za pretraživanje	20
Slika 15: Niz čvorova za brisanje predmeta	22
Slika 16: Uređivanje konfitipke za brisanje predmeta	24
Slika 17: Niz čvorova za ažuriranje predmeta.....	24
Slika 18: Forma za uređivanje predmeta.....	25