

**Diplomski studij**

**Informacijska i komunikacijska tehnologija**

**Računarstvo**

Telekomunikacije i informatika  
Obradba informacija  
Računalno inženjerstvo

**Internet stvari  
  
Sustav zalijevanja kućnih biljaka**

**Projekt**

**Paula Šalković**

**Lucija Marija Stanušić**

**Nikola Bukvić**

**Jakov Matošić**

**Leon Pongrac**

**Fabijan Bošnjak**

**Ak.g. 2023./2024.**

**Sadržaj**

[1. Uvod 2](#_Toc468366026)

[2. Opis rješenja 3](#_Toc648986165)

[3. IoT platforma 6](#_Toc1420864919)

[4. Korisnička aplikacija 11](#_Toc11399476)

[5. Raspodjela posla 15](#_Toc895639598)

# Uvod

U sklopu predmeta Internet stvari diplomskog studija na FER-u napravljena je usluga čija je svrha automatsko zalijevanje kućnih biljaka koja korisnicima omogućava precizno postavljanje gornje i donje granice vlage za svaki senzor. Trenutno, usluga obavještava korisnika kada biljka zahtijeva zalijevanje, no planirano je da usluga sama koristi aktuator za zalijevanje biljaka.

Glavni cilj usluge je automatizacija procesa zalijevanja biljaka putem senzora DHT11 za mjerenje vlage i temperature te RGB LED diode kao aktuatora za vizualno prikazivanje statusa biljaka. Sustav koristi FiPy uređaj za prikupljanje podataka od senzora i komunikaciju s platformom Home Assistant putem REST zahtjeva. Home Assistant centralizira upravljanje svim uređajima i omogućuje korisničku aplikaciju za postavljanje granica vlage i praćenje statusa biljaka. Slični sustavi uključuju Automatic Drip Irrigation Watering Kit tvrtke Raindrip, Automatic Watering System for Potted Plants tvrtke DIAFIELD i MIXC 226FT Greenhouse Micro Drip Irrigation Kit Automatic Irrigation System tvrtke MIXC.

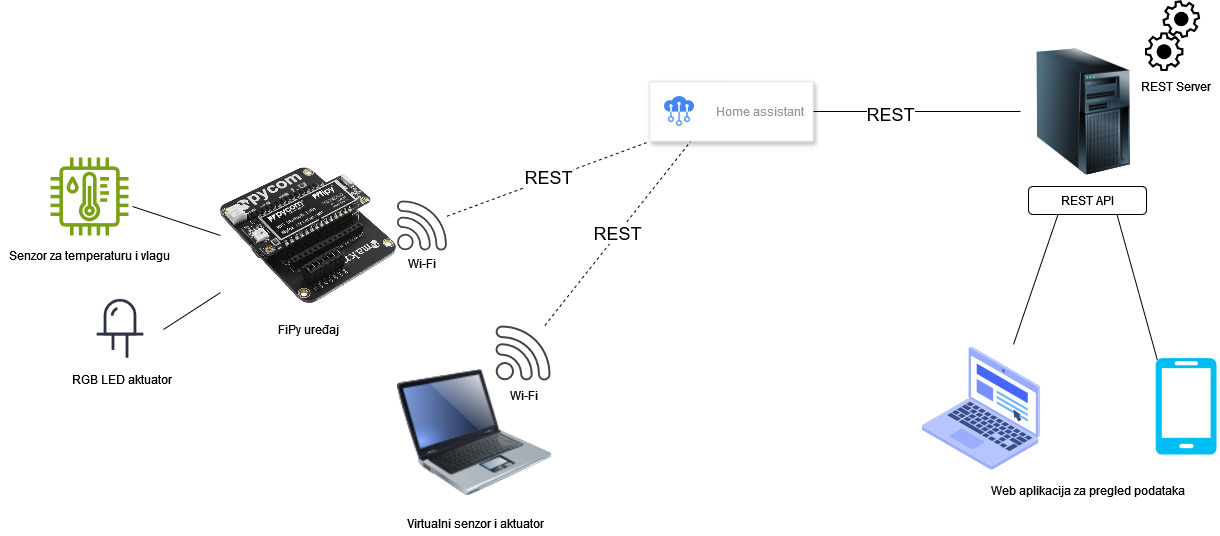
Usluga za automatsko zalijevanje biljaka ima širok spektar primjena koje pružaju korisnicima praktične prednosti i olakšavaju njihovu svakodnevicu. U privatnim domovima, ova tehnologija omogućuje korisnicima da se bezbrižno brinu o svojim biljkama čak i kada nisu kod kuće. Automatsko praćenje vlage i temperatura osigurava da biljke uvijek budu u optimalnom stanju, bez potrebe za redovitim ručnim zalijevanjem. Ovo je osobito korisno za osobe koje često putuju ili su često odsutne zbog posla ili drugih obaveza. U većim zgradama i poslovnim prostorima, automatizirano zalijevanje štedi vrijeme i resurse. Umjesto da zaposlenici ili održavatelji moraju ručno zalijevati veliki broj biljaka, sustav automatski regulira vlažnost tla i osigurava da svaka biljka dobije potrebnu količinu vode.

# Opis rješenja

Funkcionalnosti pojedinih komponenti:

* + Senzor – sakuplja informacije o temperaturi i vlazi
  + RGB LED aktuator – pomoću svoje boje pokazuje je li potrebno zalijevati biljku
  + FiPy uređaj – sakuplja informacije od senzora i šalje ih home assistantu, te prima akciju pokretanja aktuatora od home assistanta
  + Home Assistant – prati prelazi li vlaga jednu od granica i po potrebi šalje naredbu da se aktuator aktivira.
  + Korisnička aplikacija – pokazuje korisniku zadnje primljenu vlagu i temperaturu te omogućuje korisniku postavljanje granica za vlagu

Između svih mrežnih komponenti komunikacija se odvija pomoću RESTa, dok su senzor i aktuator povezani žicom s FiPy uređajem, jasniji prikaz cjelokupne arhitekture sustava prikazan je na *Slici 1*. *FiPy* uređaj koristi DHT11 senzor za mjerenje temperature i vlage, a za aktuator RGB LED diodu za vizualno prikazivanje statusa. *FiPy* uređaj koristi *Python* skriptu koja se mora objaviti na uređaj putem računala prije njegovog pokretanja. Nakon objave, uređaj može samostalno raditi bez potrebe za daljnjim povezivanjem s računalom. Skripta se može podijeliti na četiri ključna segmenta.



*Slika 1. Prikaz arhitekture razvijenog cjelokupnog sustava*

*FiPy* uređaj koristi DHT11 senzor za mjerenje temperature i vlage, a za aktuator RGB LED diodu za vizualno prikazivanje statusa. *FiPy* uređaj koristi *Python* skriptu koja se mora objaviti na uređaj putem računala prije njegovog pokretanja. Nakon objave, uređaj može samostalno raditi bez potrebe za daljnjim povezivanjem s računalom. Skripta se može podijeliti na četiri ključna segmenta.

Kako bi se omogućila komunikacija i pristup za integraciju s platformom *Home Assistant*, prvi segment koda omogućuje *FiPy* uređaju da se autentificira i poveže na lokalnu Wi-Fi mrežu koristeći IEEE 802.11 standard. Proces povezivanja započinje skeniranjem dostupnih mreža. Kod zatim uspoređuje identifikatore mreža (SSID) s listom unaprijed definiranih SSID-ova koje uređaj treba prepoznat. Nakon što identificira odgovarajuću mrežu, kod inicira autentifikaciju pomoću lozinke. Kada se uspostavi veza, završetak procesa se potvrđuje dobivanjem IP adrese, što omogućava komunikaciju unutar mreže.

Drugi segment skripte omogućuje *FiPy* uređaju da komunicira s centralnim serverom kao HTTP klijent, u ovom slučaju s *Home Assistant* platformom. Primarna funkcija uređaja kao klijenta je periodičko slanje podataka koje senzor uređaja prikupljanja. Koristeći pristupni token, uređaj se autentificira na server. S pomoću “*urequests*” biblioteke, “*send\_data\_to\_home\_assistant*” funkcija formira HTTP POST zahtjeve koje server može prihvatit i pročitat. U tijelu zahtjeva se nalaze vrijednost vlage koju očita senzor i identifikator uređaja, koji pomaže u razlikovanju podataka ako je više uređaja povezano na isti server. Nakon formiranja zahtjeva, uređaj šalje zahtjev na konfigurirani URL krajnju točku. Uređaj zatim prihvaća odgovor servera. Ako je odgovor statusni kod 200, podaci su uspješno primljeni.

Treći segment skripte implementira HTTP server na *FiPy* uređaju. Ovaj server odgovoran za primanje i obradu dolaznih HTTP zahtjeva od strane *Home Assistant* platforme. Korištenjem “*socket*” biblioteke, uređaj s pomoću “*http\_server*” funkcije postavlja HTTP server na portu 80, standardnom portu za HTTP komunikaciju. Server cijelo vrijeme čeka dolazne zahtjeve. Kada server primi zahtjev, skripta čita sadržaj zahtjeva, uključujući HTTP metode i tijelo zahtjeva. Oviseći o sadržaju koji je dobiven, uređaj mijenja boju diode (“*water*” postavlja LED na plavu, “*nowater*” na zelenu). Nakon obrade zahtjeva, server šalje nazad klijentu statusni kod 200, potvrđujući da je zahtjev uspješno primljen i obrađen. Server koristi više niti za obradu više zahtjeva istovremeno, da se smanji mogućnost blokiranja ili kašnjenja zahtjeva.

Četvrti segment predstavlja glavnu petlju. Ona je osnovna operacija *FiPy* uređaja, koordinirajući aktivnosti svih prethodnih komponenata skripte. Koristi “*\_thread*” biblioteku za upravljanje aktivnostima u isto vrijeme bez blokiranja glavne niti. Petlja se ponavlja svake 3 sekunde da se regulira frekvencija ažuriranja podataka, čime se optimizira potrošnja energije i smanjuje mrežni promet.

Radi ograničene dostupnosti opreme, uz *FiPy* uređaj su se koristile i virtualni senzori za potrebe testiranja i demonstracije funkcionalnosti cijelog sustava. Virtualni senzori su ostvareni preko *Python* skripte koje se mogu pokrenut preko bilo kojeg računala koji može pokrenut *Python* skripte. Generalna svrha ovog koda je simulirati rad i komunikaciju stvarnog *FiPy* uređaja.

Skripta koristi “*fetch\_humidity\_data*” funkciju za generiranje simuliranih vrijednosti vlage s pomoću Gaussove distribucije. Funkcija služi kao virtualni senzor vlage, koji bi u stvarnom uređaju bio fizički senzor. Vrijednost vlage se generira nasumično oko srednje vrijednosti s malom varijacijom, a zatim se zaokružuje i ograničava unutar realističnog raspona. Funkcija “*send\_humidity\_data*” redovito šalje generirane vrijednosti vlage na REST API krajnju točku, u ovom slučaju *Home Assistant*). Funkcija simulira komunikaciju između *FiPy* uređaja i *Home Assistant* platforme. Funkcije “s*end\_humidity\_data*” te “*send\_data\_to\_home\_assistant*” koja se pokreće na *FiPy* uređaju imaju velike sličnosti što se tiče same komunikacije s *Home Assistant* platforme. Kao što *FiPy* uređaj šalje svake 3 sekunde svoja očitanja tako i virtualni senzor generira i šalje nasumičnu vrijednost za vlagu svakih 3 sekunde.

Kao i pravi *FiPy* uređaj, virtualni senzori, kod nas njih tri, imaju pokrenute svaki svoj HTTP server koji osluškuju na portovima 81, 82 i 83 s pomoću funkcija “*http\_server*” i “*handle\_request*”. Ove funkcije rade isto što i “*hande\_request*” na *FiPy* uređaju, primaju i obrađuju zahtjeve od *Home Assistanta*. Za razliku od stvarnog uređaja koji mijenja boju LED diode, virtualni senzor samo ispisuje u konzoli pokrenutog koda koja bi boja LED dioda treba biti, ovisno o primljenoj naredbi (“*water*” postavlja LED na plavu, “*nowater*” na zelenu).

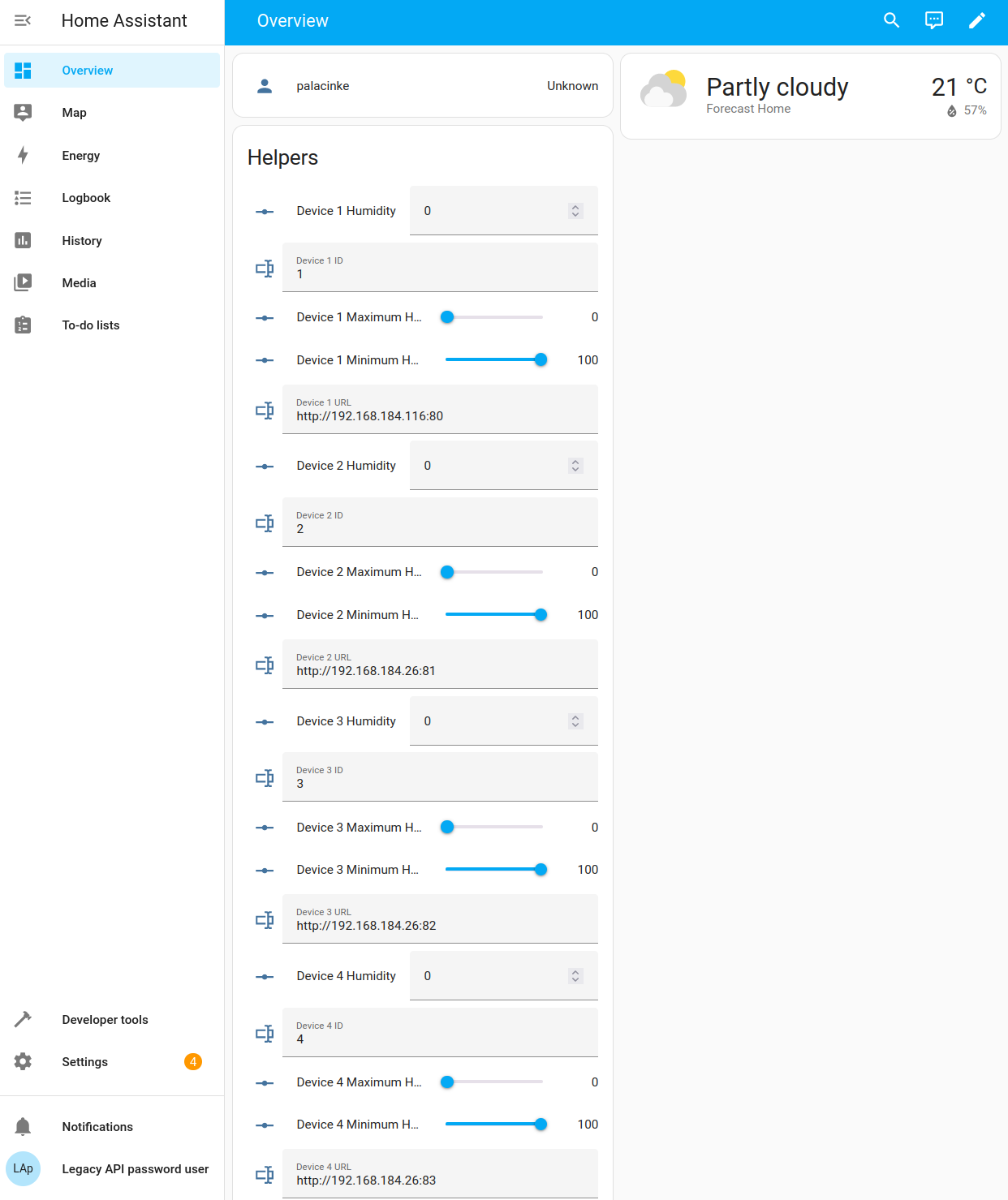
Kako bi se osiguralo kontinuirano i neometano funkcioniranje, HTTP server i funkcija za slanje podataka rade na zasebnim nitima.

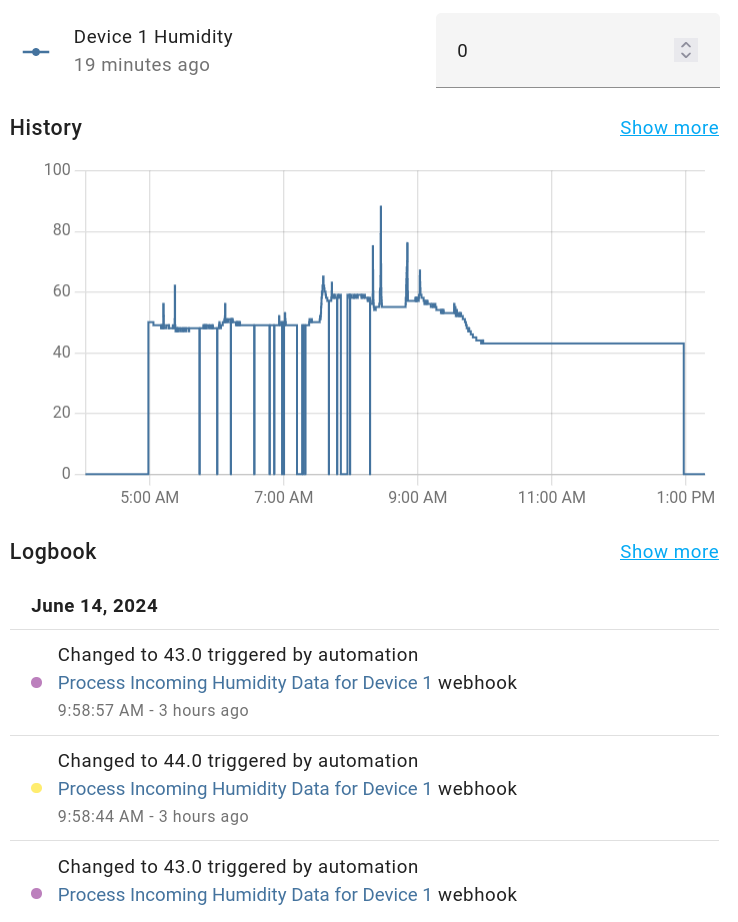
# IoT platforma

Home Assistant je open-source platforma koja omogućuje centralizirano upravljanje i automatizaciju raznih pametnih uređaja i sistema unutar kuće. Razvijen je kako bi korisnicima omogućio kontrolu nad svim pametnim uređajima, kao što su svjetla, termostati, sigurnosne kamere i mnogi drugi, s jednog mjesta. Za razliku od mnogih komercijalnih IoT platformi koje često naplaćuju korištenje putem pretplatničkih planova ili modela plaćanja po korištenju, Home Assistant nudi i inačicu besplatnu za instalaciju i korištenje, a plaćena verzija ne zahtijeva posebnu instalaciju te korisniku omogućuje usluge Home Assistant Cloud-a. Platforma se može pokrenuti na različitim uređajima, omogućava integraciju s 2000 različitih komponenata i usluga, te održava lokalnu kontrolu nad vašim pametnim domom, što znači da svi podaci ostaju unutar vaše kućne mreže. Ovo osigurava visoku razinu privatnosti i smanjuje ovisnost o vanjskim uslugama. U projektu koristimo besplatnu verziju Home Assistant-a, a za instalaciju smo odabrali Docker. Ovaj pristup pruža jednostavnost i fleksibilnost, omogućujući postavljanje Home Assistant-a na različite sustave bez potrebe za složenim konfiguracijama. Docker, kao alat za kontejnerizaciju, omogućava pokretanje aplikacije u izoliranim okruženjima, neovisnima o operacijskom sustavu.

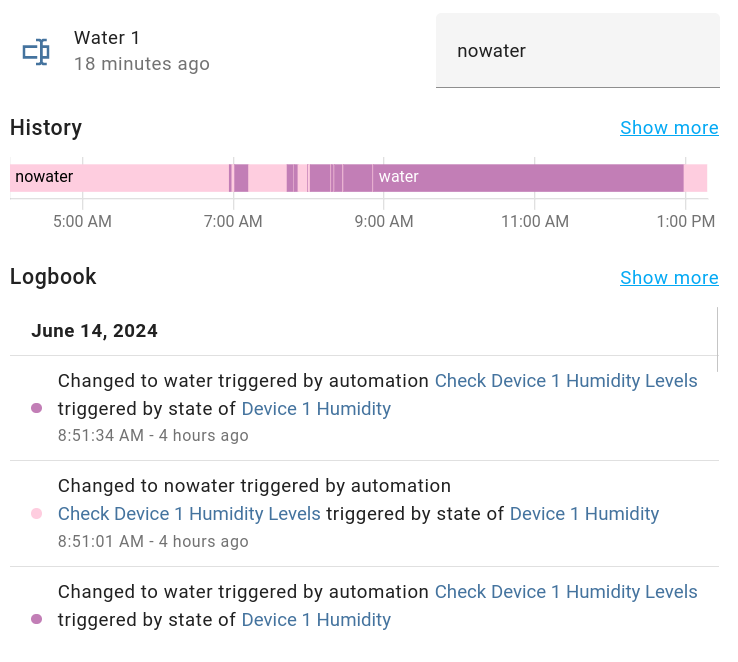
Korištenjem REST zahtjeva realizirali smo komunikaciju Home Assistant-a s ostalim dijelovima našeg sustava, web aplikacijom te virtualnim i FiPy uređajem. Početna stranica korisniku omogućava pregled nad svim povezanim uređajima tj. prikazuje statuse svakog uređaja u realnom vremenu (*Slika 2.*). Kroz projekt omogućili smo korisniku proučavanje promjene vlažnosti zemlje biljke te smo kod virtualnih senzora ostvarili simulaciju paljenja LED diode u trenutku pada vrijednosti ispod minimalno definirane granice.

Promjene u sustavu se također prikupljaju i spremaju lokalno što omogućuje detaljnu analizu. Kroz web sučelje Home Assistant-a moguće je prikazati graf promjene statusa vlažnosti zasebnih uređaja kroz vrijeme. *Slika 3.* prikazuje promijene vrijednosti vlažnosti senzora povezanog na FiPy uređaj. Detaljniji opis moguće je proučiti kroz Dnevnik (eng. Logbook) gdje su registrirane sve automatizirane promjene kojem je moguće pristupiti putem poveznice koja je vidljiva odmah ispod grafa. *Slika 4.* prikazuje promjenu stanja “water”, koja uvjetuje izmjenu boje LED lampice povezane na FIPy uređaj.

*Slika 2. Prikaz početne stranice našeg Home Assistant-a*

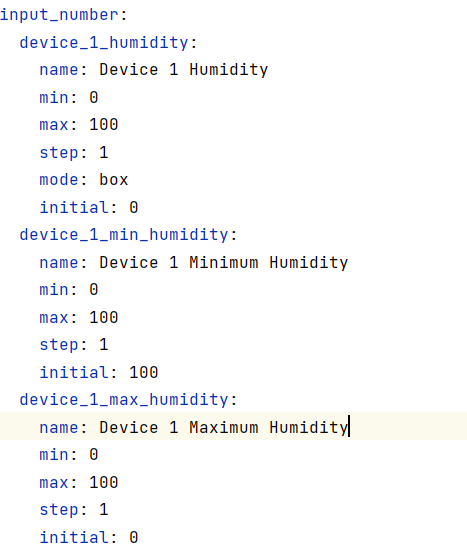


*Slika 3. Promijene vrijednosti vlažnosti senzora povezanog na FiPy uređaj*



*Slika 4. Prikaz promjene stanja polja “water”, povezanog s uređajem FiPy*

Konfiguracijski datoteka (configuration.yaml) Home Assistant-a pruža sveobuhvatan pregled kako su resursi integrirani i povezani unutar platforme. Autentifikacija sustava namještena je kroz Bearer tokene za pristup. Za spremanje i pristup podacima sa senzora definirana su polja tipa input\_number i input\_text koja omogućavaju dinamičku izmjenu podatka i interakciju (Slika 5., Slika 6.). Kako bismo poboljšali funkcionalnost sustava i olakšali nadogradnje, aktivno smo razmatrali strategije za implementaciju jednostavne skalabilnosti. Nažalost, trenutna verzija nije uspjela ispuniti te zahtjeve u potpunosti. Posljedično smo težili smanjenju krhkosti same implementacije u konfiguracijskoj datoteci.

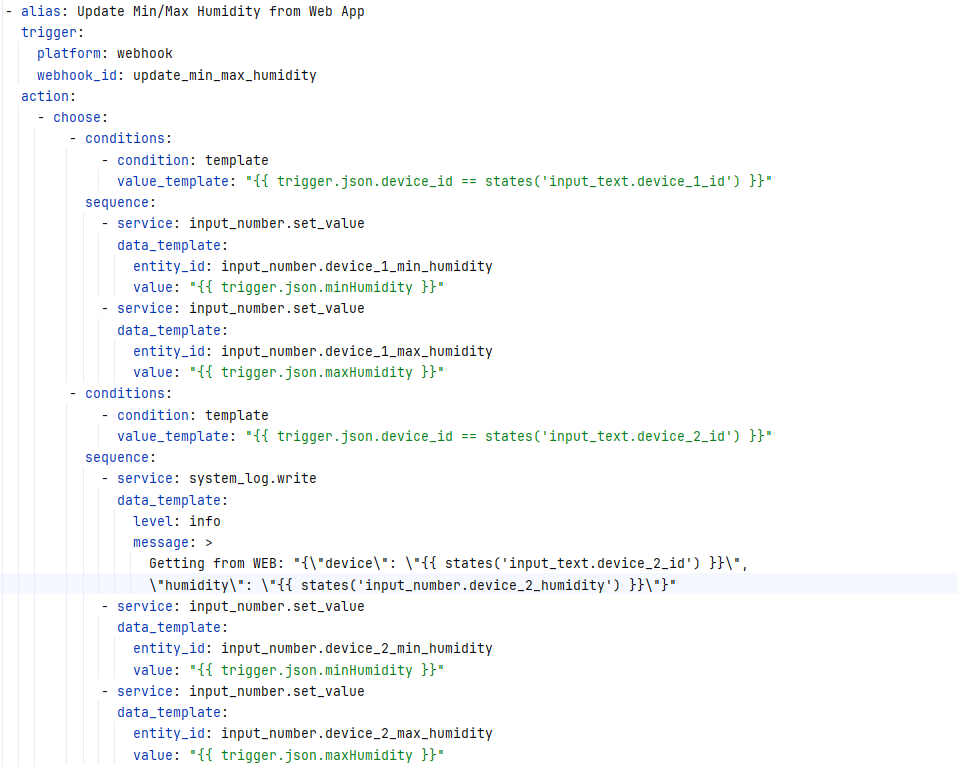


*Slika 5. i Slika 6. Definicija polja stanja uređaja FiPy*

*Slika 7. Prikaz metoda za slanje REST zahtjeva na FiPy i Web aplikaciju*

Budući da smo komunikaciju s obje strane izvršavali putem REST zahtjeva bilo je potrebno definirati odlomak rest\_command koji sadrži metode za slanje zahtjeva na za to predviđene URL adrese, kao što je prikazano na *Slici 7.*

Glavni dio svega su ipak automatizacijske skripte koje definiraju u kojem trenutku se što treba izvršiti. Webhook-ovi igraju ključnu ulogu u ovim automatizacijama, omogućavajući Home Assistant-u da reagira na događaje koji se događaju izvan njegovog okruženja. Kada određeni uređaj ili aplikacija pošalje podatke putem webhook-a, ti podaci aktiviraju specifične skripte unutar Home Assistant-a koje obrađuju i koriste te informacije za daljnje akcije. Na primjer, kada senzori vlage pošalju ažurirane podatke, webhook aktivira automatizaciju koja ažurira prikaz vlažnosti u sustavu i može pokrenuti dodatne radnje, kao što je prilagodba postavki uređaja ili slanje obavijesti (Slika 8.).



*Slika 8. Automatizacijska skripta ažuriranja vrijednosti minimalne i maksimalne vlažnosti senzora uz pomoć webhook-a*

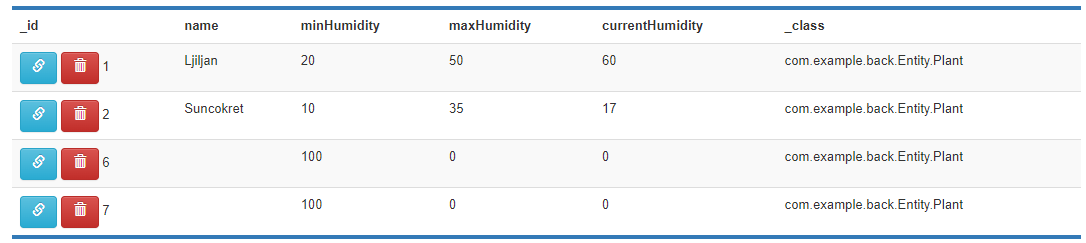
# Korisnička aplikacija

Backend dio aplikacije napravljen je u Java Spring Bootu te koristi MongoDB bazu podataka za zapisivanje detalja o biljkama, odnosno uređajima. Backend dio aplikacije s pomoću REST-a komunicira s Home Assistantom i frontendom aplikacije. Od Home Assistanta prima podatke kada se pojavi novi uređaj te ga zapiše u bazu te ga čuva dok korisnik u aplikaciji ne traži nove uređaje kojima bi dodijelio vrijednosti imena biljke i minimalne i maksimalne vlage koju biljka tolerira. Nakon što korisnik upiše te podatke, oni se direktno šalju na Home Assistant koji ih zapiše.

Backend je napravljen u obliku MVC-a (Model, View, Controller) koristeći Springove anotacije @Controller, @Service i @Repository za jednostavno primanje i slanje GET i POST zahtjeva te ubacivanje i dohvaćanje podataka iz MongoDB baze. Baza podataka je takozvana NO-SQL, odnosno sprema JSON-like objekte u tablicu Plant. Novo dodani objekt će u bazi imati nedefinirane minimalne i maksimalne vrijednosti vlažnosti te trenutnu vlažnost koju će kasnije poprimiti kroz razgovor s Home Assistantom.

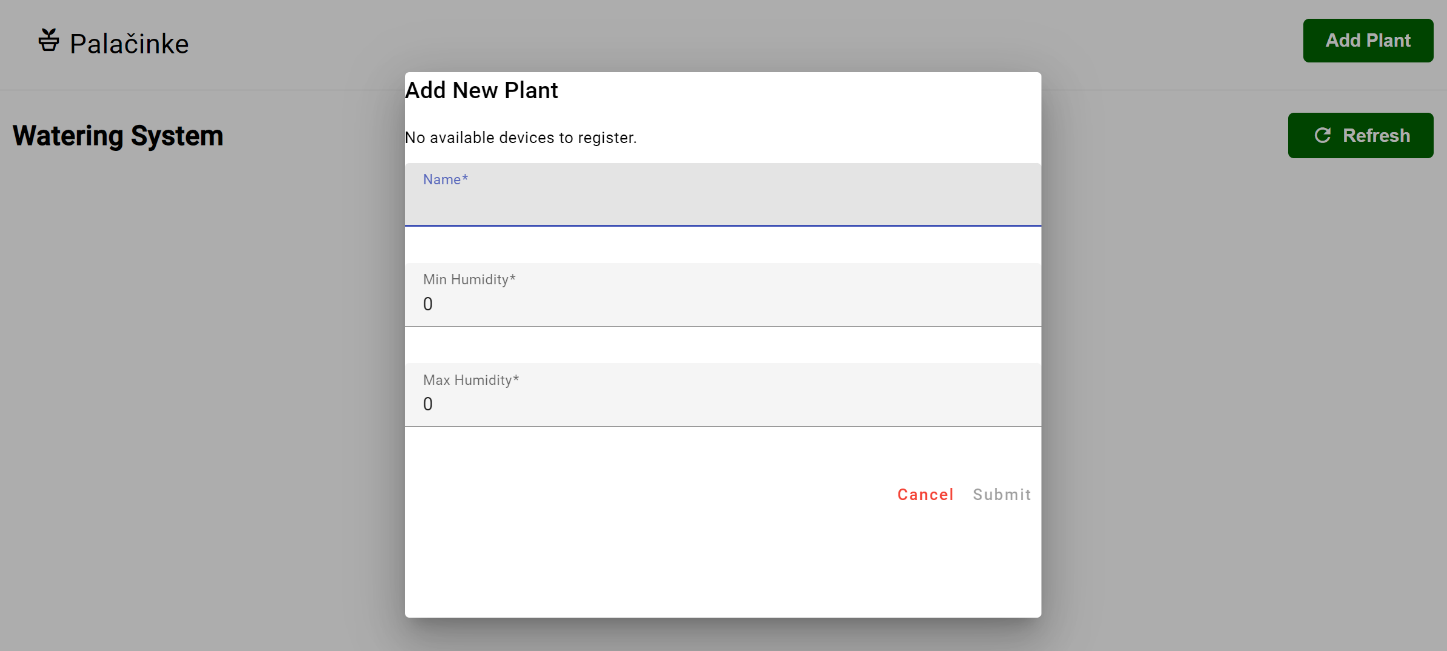


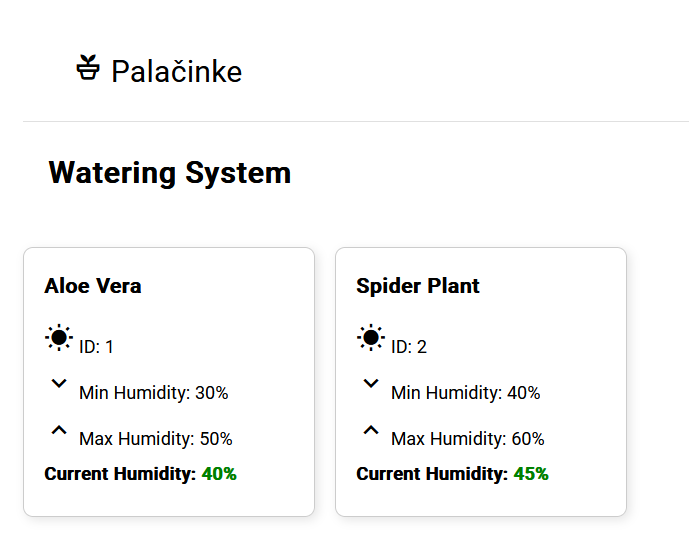
*Slika 9. Objekt Plant koji definira objekt za bazu podataka*

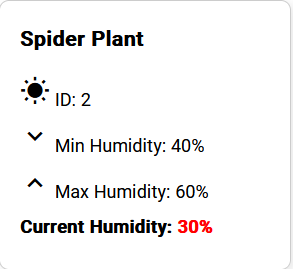
*Slika 10. Baza podataka za biljke u MongoDB*

Na slici 10, prva dva ubačena objekta u bazu ubačena su kroz aplikaciju, dok su ostala dva – s vrijednostima id-a 6 i 7 - još neubačena u aplikaciju i ponude se korisniku da im unese vrijednost. Definicija objekata u Spring Bootu nalazi se na slici 9.

Frontend dio aplikacije razvijen je na sljedeći način: U glavnom sučelju, korisnici mogu vidjeti sve svoje biljke, s relevantnim informacijama kao što su ID biljke, minimalna i maksimalna vlažnost tla, te trenutna vlažnost. Funkcionalnost aplikacije uključuje mogućnost uređivanja već postojećih uređaja putem jednostavnog obrasca koji se otvara klikom na gumb "Add Plant"(*Slika 11.*). U obrascu, korisnici mogu odabrati ID iz niza neregistriranih uređaja, unijeti ime biljke te postaviti minimalnu i maksimalnu vlažnost tla. Ako nema dostupnih ID-ova, prikazuje se poruka upozorenja. Osim toga, aplikacija omogućava korisnicima osvježavanje podataka o trenutnoj vlažnosti tla za svaku biljku klikom na gumb "Refresh". Vizualni pokazatelji u aplikaciji (zelena i crvena boja) pomažu korisnicima da brzo procijene je li trenutna vlažnost tla unutar željenog raspona, čime se olakšava praćenje i održavanje optimalnih uvjeta za rast biljaka (*Slika 12., Slika 13.*).

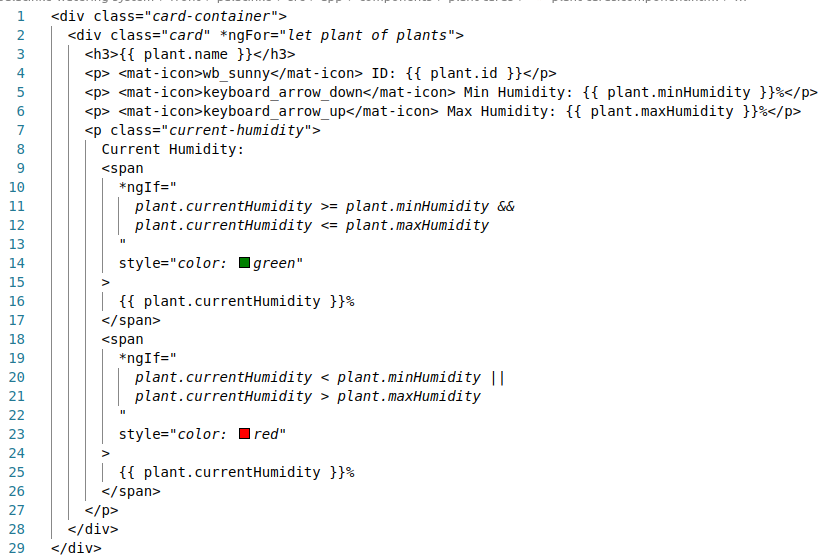
*Slika 11. Forma za dodavanje nove biljke.*

*Slika 12. Prikaz biljaka i trenutne vlage.*



*Slika 13. Primjer preniske vlage*

Angular je iskorišten u aplikaciji kako bi se omogućila modularnost i komponentno orijentirani pristup razvoju. Korištenje reaktivnih formi (eng. Reactive Forms) omogućava jednostavno upravljanje i validaciju obrazaca (*Slika 14.*). Angular Material je korišten za izgradnju modernog i responzivnog korisničkog sučelja, uključujući elemente kao što su mat-form-field, mat-select, mat-button, i mat-icon. Servisi u Angularu, kao što je PlantService, omogućuju pristup podacima putem HTTP zahtjeva, dok su pretplate na Observable korištene za asinkrono rukovanje podacima.

*Slika 14. Generiranje komponenti Angular direktivama*

# Raspodjela posla

*Dokumentacija:*

* *Opis rješenja: Leon Pongrac (uvodni dio) i Nikola Bukvić (opis rada koda)*
* *IoT platforma: Lucija Marija Stanušić (opis platforme i sustava) i Paula Šalković (objašnjenje konfiguracije)*
* *Korisnička aplikacija: Fabijan Bošnjak (Backend) i Jakov Matošić (Frontend)*

*Uređaj:*

* *Implementacija:*
  + *Pravi uređaj: Leon Pongrac i Nikola Bukvić*
  + *Virtualni uređaj: Nikola Bukvić*

*IoT platforma:*

* *Konfiguracija HA: Paula Šalković i Lucija Marija Stanušić*
* *Integracija IoT sustava: Lucija Marija Stanušić i Paula Šalković*

*Korisnička aplikacija:*

* *Implementacija:*
  + *Frontend: Jakov Matošić*
  + *Backend: Fabijan Bošnjak*