

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 000

**PRAĆENJE VOZILA JAVNOG GRADSKOG
PRIJEVOZA**

Nikolina Mijoč

Zagreb, lipanj 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 000

**PRAĆENJE VOZILA JAVNOG GRADSKOG
PRIJEVOZA**

Nikolina Mijoč

Zagreb, lipanj 2019.

Sadržaj

1. Uvod.....	2
2. Usluga suradnog opažanja okoline.....	3
3. Interoperabilnost sustava i SymbloTe programski okvir.....	6
3.1. SymbloTe Cloud.....	8
3.2. SymbloTe Core.....	10
3.3. Konfiguracija platforme te instalacija SymbloTe Cloud komponenti.....	11
3.3.1 Konfiguracija SymbloTe platforme.....	13
3.3.2 Postavljanje SymbloTe Cloud komponenti.....	14
4. Primjena usluge suradnog opažanja okoline i SymbloTe programskog okvira u izradi rješenja za praćenje vozila javnog gradskog prijevoza.....	17
4.1. Struktura i pohrana podataka.....	17
4.2. Registracija resursa na SymbloTe Core.....	18
4.3. RAP Plugin.....	22
5. Zaključak.....	27
6. Literatura.....	28

1. Uvod

Internet stvari (engl. *Internet of Things*, skraćeno IoT) je koncept povezivanja virtualnih i stvarnih objekata na Internet te njihovo međusobno povezivanje. Uređaji putem raznih ugrađenih senzora prikupljaju podatke te ih šalju na IoT platforme s kojima su povezani. Glavne uloge IoT platformi su omogućavanje interoperabilnosti, pružanje podrške programima na uređajima, njihovo povezivanje, osiguranje sigurnosti te prikupljanje, vizualizacija i analiza podataka [1]. *Symbiosis of smart objects across IoT environments* (skraćeno SymbloTe) je programski okvir za suradnju IoT platformi. Unificirano pronalazi i koristi umrežene uređaje, senzore i aktuatora, koji su u različitim administrativnim domenama i platformama, radi jednostavnijeg razvoja novih mobilnih i web korisničkih usluga. Također, nudi i mogućnost kreiranja vlastite IoT platforme te je korišten prilikom izrade ovog rada kako bi programsko rješenje za praćenje vozila javnog gradskog prijevoza imalo sva prethodno navedena obilježja.

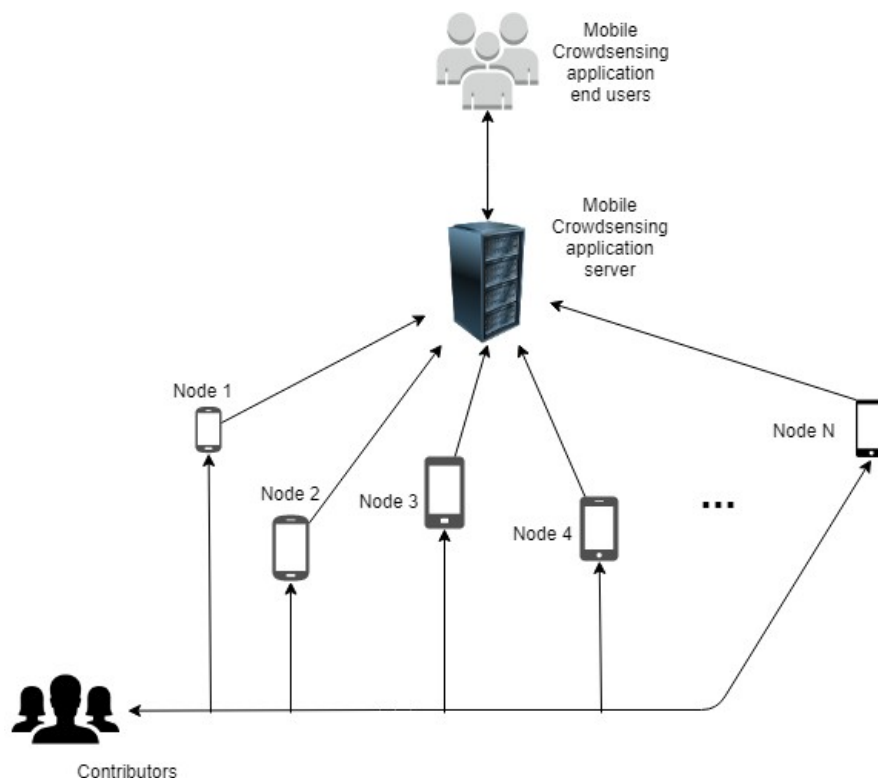
Trenutno, na tržištu ne postoji mobilna aplikacija, za područje grada Zagreba, koja bi korisnicima omogućila prikaz stvarnih vremena dolazaka vozila Zagrebačkog električnog tramvaja (skraćeno ZET) na pojedinu stanicu. Razlog tome je taj što odbijaju javno podijeliti GPS lokacije svojih vozila i time ostaju jedini pružatelji informacija. Podatke o stvarnom vremenu dolazaka vozila pruža isključivo ZET-ov sustav, odnosno uređaji postavljeni na svakoj stanici, kojem su nedostaci netočnost podataka te da su ljudi su primorani fizički doći do pojedine stanice da bi saznali iste.

Cilj ovog rada je iskoristiti koncept Interneta stvari u svrhu izrade aplikacije za praćenje vozila javnog gradskog prijevoza pomoću SymbloTe programskog okvira. Aplikacija dohvaća podatke o lokaciji i broju putnika vozila te na zahtjev korisnika računa vrijeme dolaska tramvaja na određenu stanicu. Podaci prikupljeni mobilnim telefonima korisnika uslugom suradnog opažanja okoline prenose se na SymbloTe platformu čime se osigurava njezina interoperabilnost te dostupnost i pouzdanost informacija.

2. Usluga suradnog opažanja okoline

Naglo povećanje broja mobilnih uređaja te njihov razvoj u kontekstu procesiranja, ugradnje senzora i prostora za pohranu imalo je presudan utjecaj na pojavu usluge suradnog opažanja okoline (engl. *mobile-crowdsensing*). Ideja iza ove tehnike je da mobilni uređaji putem svojih senzora prikupljaju određene podatke iz svog okruženja poput osvjetljenja, zvukova, lokacije, brzine te ih efektivno koriste u sustavima za praćenja kvalitete zraka, stanja u prometu, razine buke i tako dalje. Time ova usluga postaje dobra alternativa tradicionalnim stacionarnim postajama pri čemu mali pokretni senzori distribuiraju podatke skupini ljudi i time doprinose cijelom sustavu.

Aplikacije temeljene na usluzi suradnog opažanja okoline obično rade tako da se senzorski podaci očitani putem mobilnog uređaja prenose putem bežične mreže na poslužitelje gdje se dalje obrađuju, analiziraju te prikazuju korisnicima koji zatraže tu informaciju (slika 1).



Slika 1: Usluga suradnog opažanja okoline

Mobilna komponenta mora biti sposobna prikupljati parametre okoline, prenoseći ih prema središnjem poslužitelju. Postupak prikupljanja podataka temelji se na senzorima pametnih telefona ili na malim vanjskim senzorima koji su dostupni putem pametnog telefona, a proces prijenosa obično se oslanja na povezivanje pametnog telefona s Internetom. Unatoč delegiranju prijenosnih zadataka na pametne telefone, vanjski senzorski uređaji moraju imati mogućnosti komunikacije za prijenos prikupljenih podataka na pametni telefon. Dakle, senzorski uređaj treba biti opremljen bežičnim komunikacijskim sučeljem. Središnji procesni poslužitelj mora moći primiti prenesene podatke sa senzora, pohranjivati i obrađivati podatke, te ispravno prezentirati dobivene rezultate upraviteljima sustava. Osim toga, u nekim slučajevima, središnji poslužitelj obavlja daljinsku komunikaciju s mobilnim uređajima za konfiguracijske zadatke, čime se omogućava dinamičko mijenjanje ponašanja osjetila.

Suradno opažanje okoline rezultiralo je razvojem mnogobrojnih korisničkih aplikacija koje se dijele u dvije kategorije:

- Aplikacije fokusirane na korisnika (engl. *People-centric applications*)
- Aplikacije fokusirane na okruženje (engl. *Environment-centric applications*)

Aplikacije fokusirane na korisnika uglavnom prikupljaju senzorske podatke vezane uz samog pojedinca. Prikupljene informacije dostupne su samom korisniku, a mogu biti iskorištene u kombinaciji s drugim korisničkim podacima u svrhu cijele zajednice. Aplikacije fokusirane na okruženje, za razliku od onih fokusiranih na korisnika, koriste podatke koje su od važnosti cijeloj zajednici. Paradigma pametnog grada koristi ovakav tip aplikacija za prikupljanje informacija u svrhu učinkovitijeg korištenja prometne infrastrukture, izračunavanja kvalitete zraka, razine buke i tako dalje.

S obzirom na to da je usluga suradnog opažanja temeljena na mrežnoj i senzorskoj infrastrukturi, ključne izazove predstavljaju prijenos podataka, njihova krajnja analiza i procesiranje, procjena pouzdanosti istih te očuvanje korisnikove privatnosti i energije uređaja [2].

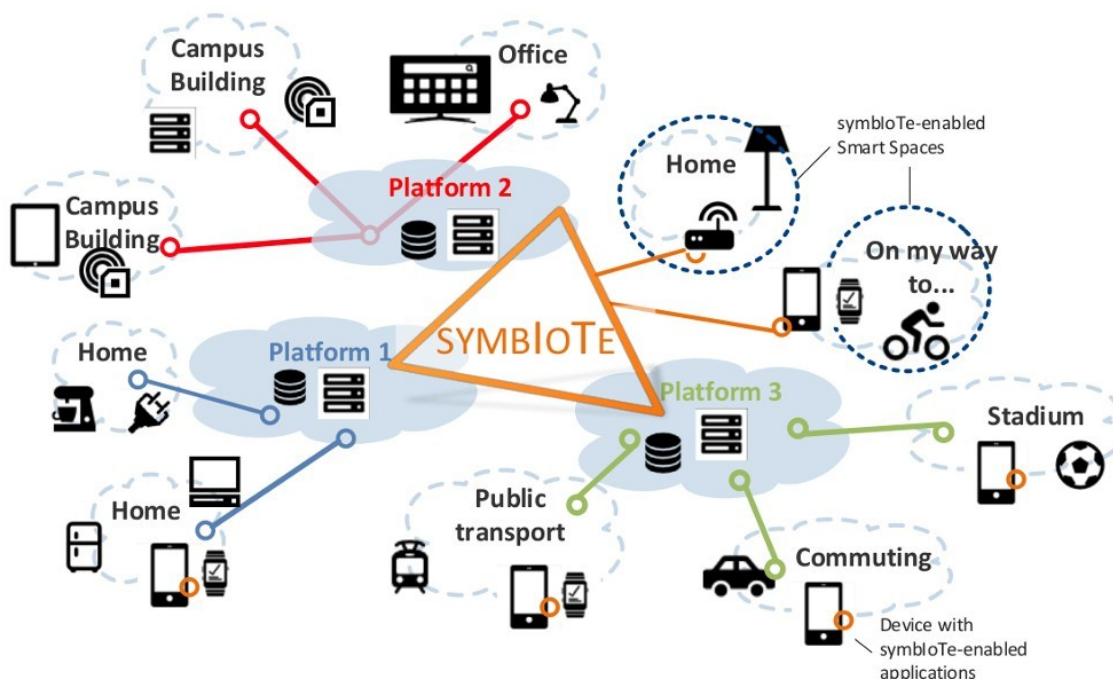
Klasifikacija suradnog opažanja okoline ovisna o interakciji korisnika je podijeljena u dva tipa:

- Participativno prikupljanje – korisnici dobrovoljno sudjeluju u prikupljanju informacija
- Oportunističko prikupljanje – podaci očitani sa senzora su pozadinski prikupljeni bez direktne intervencije korisnika

U sklopu ovog rada, podaci o lokaciji korisnika koji se vozi tramvajem ZET-a, prikupljeni uslugom suradnog opažanja, korišteni su za izračun vremena dolaska tramvaja na određenu stanicu. Osim vremena dolaska vozila na određenu stanicu, participativnim prikupljanjem izračunat je broj osoba u pojedinom vozilu. Informacije se na zahtjev prenose na SymbloTe platformu gdje su vizualno prikazane te dostupne drugim SymbloTe platformama.

3. Interoperabilnost sustava i SymbloTe programski okvir

Tijekom nekoliko posljednjih godina napredak tehnologije, povećanje broja uređaja povezanih na Internet te razvitak koncepta Interneta stvari doveli su do razvitka velikog broja vertikalnih rješenja. Različite izvedbe postignute su u pogledu konfiguracije samih uređaja, programskih sučelja aplikacija (engl. *Application Programming Interface*, skraćeno API), formata pohrane podataka te same IoT infrastrukture. Da bi se postigao cilj IoT-a, a taj je da svi uređaji spojeni na Internet međusobno komuniciraju, potrebno je postići interoperabilnost sustava, odnosno omogućiti suradnju sustava različitih izvedbenih rješenja. SymbloTe nudi rješenje za unificirani pogled na različitim platformama i njihovim resursima, tako da su resursi platforme transparentni dizajnerima te programerima aplikacija. Osim toga, omogućava implementaciju federacija IoT platformi kako bi mogle međusobno djelovati sigurno, dijeliti resurse na obostranu korist te podržati migraciju pametnih objekata između različitih domena i platformi Interneta stvari (slika 2) [3].

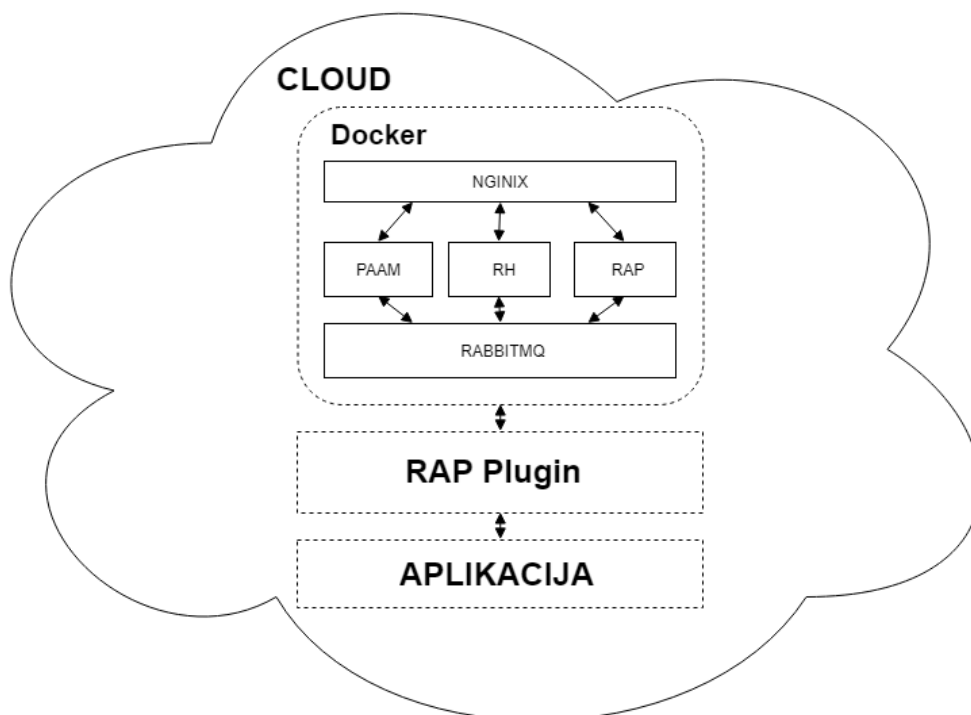


Slika 2: Interoperabilnost SymbloTe programskog okvira (preuzeto iz [3])

3.1. SymbloTe Cloud

SymbloTe Cloud je “mozak” programskog okvira SymbloTe. Podaci koji se prikupe putem raznih senzora šalju se na računalni oblak (engl. *Cloud*) gdje se obrađuju te koriste u razne analitičke svrhe. Komponente od kojih se sastoji su sljedeće:

- Upravitelj autentifikacije i autorizacije platforme (engl. *Platform Authentication and Authorization Manager*, skraćeno PAAM) – omogućava sigurnu i pouzdanu vezu tijekom razmjene resursa
- Upravljač registracije (engl. *Registration Handler*, skraćeno RH) – upravlja registracijom, osvježavanjem te brisanjem resursa
- Posrednik pristupa resursima (engl. *Resource Access Proxy*, skraćeno RAP) – krajnja točka pristupa resursima; zadužen je za dohvat podataka prikupljenih senzorima ili aktivaciju samih senzora [4]

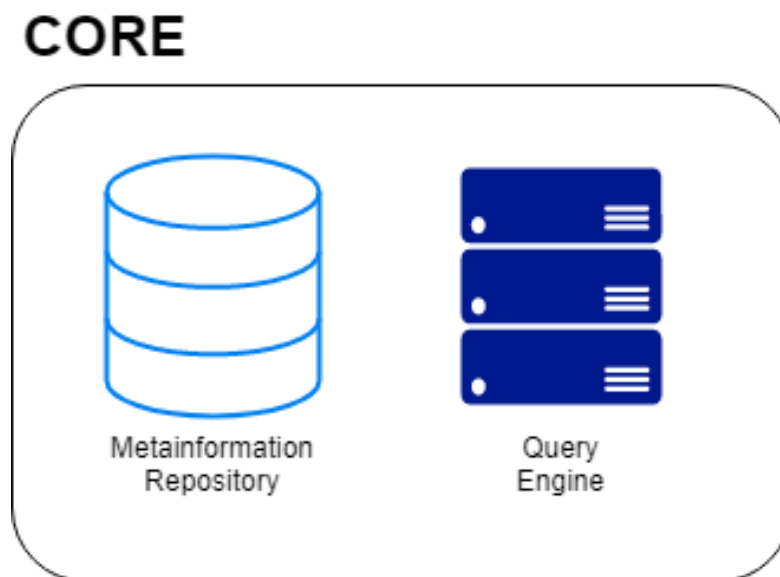


Slika 3: Komponente SymbloTe Cloud-a

Docker je platforma otvorenog koda koja omogućuje virtualizaciju na razini OS-a, a koristi se za razvoj, implementaciju i pokretanje distribuiranih aplikacija te svih njenih ovisnosti u obliku Docker kontejnera. SymbloTe Cloud komponente pokreću se preko Dockera, što će biti prikazano u sklopu ovog rada, ili direktno na uređajima s Linux operacijskim sustavom. Navedene komponente proširuju aplikaciju interoperabilnim mogućnostima te pomoću RAP Plugin-a osiguravaju komunikaciju, prijenos i dohvaćanje podataka.

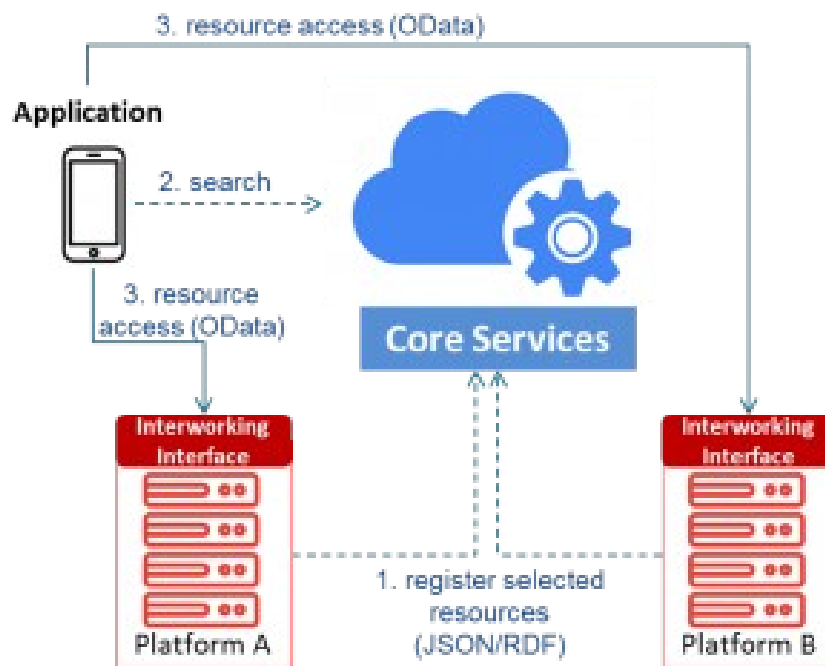
3.2. SymbloTe Core

SymbloTe Core je centralna točka na kojoj aplikacije različitih programskih sučelja mogu pretraživati resurse različitih platformi. Dvije glavne sastavnice Core servisa su spremnik metapodataka resursa te mehanizam za upite (slika 4).



Slika 4: Komponente SymbloTe Core servisa

IoT platforme registriraju isključivo metapodatke resursa na Core servise, a zahtjevi za pristup resursima usmjereni su na odgovarajuću platformu. Po završetku provjere identiteta korisnika aplikacije, mehanizam za upite obrađuje upit prema kojem se u repozitoriju metapodataka traže metapodaci određenog resursa. Korisničkoj aplikaciji vraća se URL s pristupom resursu (slika 5).



Slika 5: Pristup resursima na SymbloTe Core-u (preuzeto iz [5])

3.3. Konfiguracija platforme te instalacija SymbloTe Cloud komponenti

Rezultat sljedećih koraka je postavljanje i pokretanje SymbloTe Cloud komponenti za vlastitu platformu te registracija same platforme i resursa na SymbloTe Core. Navedeni postupak omogućava drugim SymbloTe korisnicima pretraživanje i pristup resursima dijeljenih izrađenom platformom. Pokretanje komponenti moguće je pomoću *ngrok* alata, što će biti prikazano u sklopu ovog rada, ili pomoću javne IP adrese s DNS ulazom. Potrebno se registrirati te instalirati *ngrok* klijenta koji omogućuje lokalnom poslužitelju zaobilazak NAT-a, vatrozida te pristup javnom internetu preko sigurnih tunela. Instalacija *ngrok* alata obavlja se registracijom te dohvaćanjem *zip* datoteke dostupne putem poveznice [6]. Datoteku je nužno raspakirati te, naposljetku, povezati s kreiranim računom putem tokena za provjeru identiteta koji je dobiven prilikom registracije (slika 6).

```

+ symbiote wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
--2019-05-29 12:20:39-- https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
Resolving bin.equinox.io (bin.equinox.io)... 52.73.9.93, 52.71.139.107, 52.3.53.115, ...
Connecting to bin.equinox.io (bin.equinox.io)[52.73.9.93]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 16648024 (16M) [application/octet-stream]
Saving to: 'ngrok-stable-linux-amd64.zip'

ngrok-stable-linux-amd64.zip      100%[=====] 15,88M  8,92KB/s  in 1,8s

2019-05-29 12:20:41 (8,92 MB/s) - 'ngrok-stable-linux-amd64.zip' saved [16648024/16648024]

+ symbiote ll
total 16M
-rw-rw-r-- 1 student student 16M svi 29 12:20 ngrok-stable-linux-amd64.zip
+ symbiote unzip ngrok-stable-linux-amd64.zip
Archive:  ngrok-stable-linux-amd64.zip
  inflating: ngrok
+ symbiote ll
total 50M
-rwxr-xr-x 1 student student 34M svi 19 12:07 ngrok
-rw-rw-r-- 1 student student 16M svi 29 12:20 ngrok-stable-linux-amd64.zip
+ symbiote ./ngrok authtoken 43NRAH1RwsiUhsSaktByYV NFdFZgASgDc46KNoUKoB
Auth token saved to configuration file: /home/student/.ngrok2/ngrok.yml

```

Slika 6: Instalacija ngrok klijenta te povezivanje s korisničkim računom

Pokretanje tunela obavlja se naredbom „./ngrok http –bind-tls „true“ 8102“ kojoj prethodi GNU screen naredba „screen -S ngrok“ koja omogućava da ngrok klijent ostane pokrenut i prilikom odjave. Rezultat komandi je tunel pokrenut na vratima 8102 (slika 7).

```

ngrok by @inconshreveable

Session Status      online
Account             Nikolina Mijoc (Plan: Free)
Version             2.3.29
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           https://ebd8ccb8.ngrok.io -> http://localhost:8102

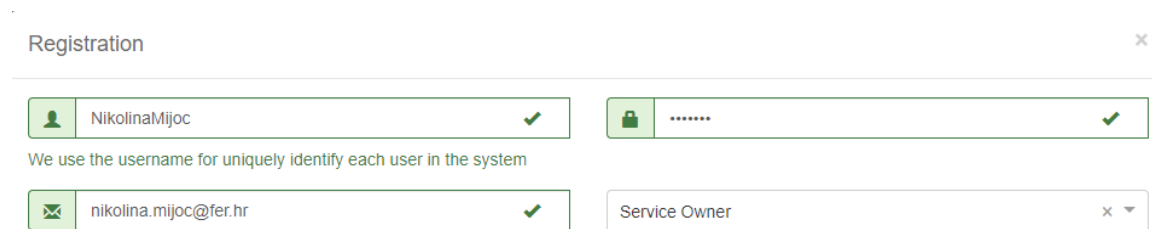
Connections         ttl      opn      rt1      rt5      p50      p90
                    8761     0        0.04     0.04     9.53     9.73

```

Slika 7: Ngrok tunel pokrenut pomoću GNU screen naredbe

3.3.1 Konfiguracija SymbloTe platforme

Izrada same platforme napravljena je na SymbloTe Core Admin stranici, dostupnoj na poveznici [7], kojoj prethodi registracija računa. Tijekom registracije potrebno je osigurati korisničko ime, lozinku, adresu e-pošte te ulogu koja je u ovom slučaju „Service Owner“ (slika 8).



The screenshot shows a 'Registration' form with a close button (X) in the top right corner. The form contains four input fields, each with a green checkmark indicating successful validation:

- Username: 'NikolinaMijoc' (with a person icon on the left)
- Password: '*****' (with a lock icon on the left)
- Email: 'nikolina.mijoc@fer.hr' (with an envelope icon on the left)
- Role: 'Service Owner' (with a dropdown arrow on the right)

Below the username field, there is a note: 'We use the username for uniquely identify each user in the system'.

Slika 8: Registracija na SymbloTe Core Admin stranicu

Na unesenu adresu e-pošte dostavljena je poveznica čijim se odabirom aktivira izrađeni korisnički račun. Po završetku prijave ponuđena je opcija registracije nove platforme te slijedno tome potrebno je popuniti sljedeće podatke:

- Identifikator platforme
- Ime platforme
- Opis platforme
- URL usluga za međudjelovanje
- Informacijski model međudjelovanja sučelja
- Tip platforme

Unutar polja gdje se traži URL usluga za međudjelovanje, potrebno je staviti *ngrok* URL dobiven u potpoglavlju 3.3. Odabirom opcije „Submit“ platforma je registrirana u SymbloTe Core (slika 9).

Platform Id <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;">yplatform ✓</div> <p>From 4 to 30 characters. Include only letters, digits, '-' and '_'. You can leave it empty for autogeneration</p>	Platform Name <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;">Y_Platform ✓</div> <p>From 3 to 30 characters</p>
--	---

Platform Descriptions

Platforma za praćenje vozila javnog gradskog prijevoza. ✓

+ -

From 4 to 300 characters

Interworking Services

https://ebd8ccb8.ngrok.io/ ✓

BIM × ▼

Type

Platform ▼

Select your Platform type

Submit Close

Slika 9: Registracija platforme

3.3.2 Postavljanje SymbloTe Cloud komponenti

Tri su načina na koje je moguće instalirati Cloud komponente:

- 1) Postavljanjem izvornih komponenti
- 2) Preuzimanjem i konfiguracijom jar datoteka
- 3) Preuzimanjem i konfiguracijom Docker kontejnera

Za potrebe izrade ovog rada odabrana je treća opcija, to jest instalacija putem preuzimanja i konfiguracije Docker kontejnera. Kako bi se to ostvarilo, neophodno je preuzeti sljedeće alate:

- Docker – verzija 18.03.x
- Docker-compose – verzija 1.21.x

Upute za instalaciju Docker alata dostupne su na stranicama samog alata, na poveznici pod [8], te su u ovom radu korištene za postavljanje. Proces se sastoji od izvođenja niza naredbi (slika 10) čiji je rezultat omogućena funkcionalnost alata.

```
root@student-HVM-domU:~# sudo apt-get update
Hit:1 https://download.docker.com/linux/ubuntu bionic InRelease
Reading package lists... Done
root@student-HVM-domU:~# sudo apt-get install \
> apt-transport-https \
> ca-certificates \
> curl \
> software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
apt-transport-https is already the newest version (1.6.10).
curl is already the newest version (7.58.0-2ubuntu3.7).
ca-certificates is already the newest version (20180409).
software-properties-common is already the newest version (0.96.24.32.8).
The following packages were automatically installed and are no longer required:
 aufs-tools cgroupfs-mount containerd.io docker-ce-cli pigz
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
root@student-HVM-domU:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
OK
root@student-HVM-domU:~# sudo apt-key fingerprint 0EBFCD88
pub  rsa4096 2017-02-22 [SCEA]
    9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid          [ unknown] Docker Release (CE deb) <docker@docker.com>
sub  rsa4096 2017-02-22 [S]

root@student-HVM-domU:~# sudo add-apt-repository \
> "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
> $(lsb_release -cs) \
> stable"
WARNING:root:could not open file '/etc/apt/sources.list'

WARNING:root:could not open file '/etc/apt/sources.list'

Hit:1 https://download.docker.com/linux/ubuntu bionic InRelease
Reading package lists... Done
```

Slika 10: Instalacija Docker alata

Nakon registracije platforme na SymbloTe Core omogućeno je dohvaćanje komprimirane konfiguracijske datoteke koju je nužno preuzeti te otpakirati u željeni radni direktorij. Također, u sklopu dokumentacije projekta SymbloTe dostupne su docker-compose datoteke za željenu razinu prevođenja koja može biti L1 ili L2. U sklopu ovog rada dohvaćene su *yaml* datoteke za L1 integraciju dostupne na poveznici pod [9]. Izvođenjem „*docker swarm init*“ naredbe (slika 11) čvor postaje swarm upravitelj, tj. omogućen mu je enkriptirani prijenos podataka.

```

➔ ~ docker swarm init
Swarm initialized: current node (wnq4svorei9yno9c47k001ecb) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-4h5xskxqscbc910gaopfh2er38mpn2d8rbbxsic3bau0wk6lx6-e8ozqytdfry842dg5yaz9s7lh 161.53.19.196:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

```

Slika 11: Rezultat "docker swarm init" komande

Sljedeći korak u konfiguraciji je pokretanje *docker stack* kolekcije servisa. Pokretanje se radi izvođenjem naredbe „*docker stack deploy*“ s pripadajućim stack datotekama, odnosno *docker-compose yml* datotekama koje su prethodno preuzete (slika 12).

```

➔ symbiote docker stack deploy -c docker-compose-swarm-L1.yml -c docker-compose-ngrok-swarm-L1.yml -c docker-compose-swarm-custom.yml symbiote-app
WARN[0000] ignoring IP-address (127.0.0.1:8080:8080/tcp) service will listen on '0.0.0.0'
WARN[0000] ignoring IP-address (127.0.0.1:8001:8001/tcp) service will listen on '0.0.0.0'
WARN[0000] ignoring IP-address (127.0.0.1:5671:5671/tcp) service will listen on '0.0.0.0'
WARN[0000] ignoring IP-address (127.0.0.1:5672:5672/tcp) service will listen on '0.0.0.0'
WARN[0000] ignoring IP-address (127.0.0.1:15672:15672/tcp) service will listen on '0.0.0.0'
WARN[0000] ignoring IP-address (127.0.0.1:8888:8888/tcp) service will listen on '0.0.0.0'
WARN[0000] ignoring IP-address (127.0.0.1:8102:8102/tcp) service will listen on '0.0.0.0'
Creating network symbiote
Creating secret symbiote-app_nginxngrok
Creating secret symbiote-app_nginxprod
Creating secret symbiote-app_monitorbootstrapconfig
Creating secret symbiote-app_eurekabootstrapconfig
Creating secret symbiote-app_aambootstrapconfig
Creating secret symbiote-app_aamcertconfig
Creating secret symbiote-app_cloudconfigconfig
Creating secret symbiote-app_rapbootstrapconfig
Creating secret symbiote-app_rhbootstrapconfig
Creating secret symbiote-app_rpeconfig
Creating service symbiote-app_symbiote-aam
Creating service symbiote-app_symbiote-cloudconfig
Creating service symbiote-app_symbiote-monitoring
Creating service symbiote-app_symbiote-rabbitmq
Creating service symbiote-app_symbiote-rap
Creating service symbiote-app_symbiote-rpe
Creating service symbiote-app_symbiote-eureka
Creating service symbiote-app_symbiote-mongo
Creating service symbiote-app_symbiote-nginx
Creating service symbiote-app_symbiote-rh
➔ symbiote █

```

Slika 12: Rezultat naredbe "docker stack deploy" s pripadajućim stack datotekama

Rezultat prethodne naredbe su pokrenuti *docker* servisi. Nužno je pričekati dok broj replika za svaki servis ne bude različit od nule (slika 13).

```

+ symbiote docker service ls

```

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
0z06n6thbvwzhz	symbiote-app_symbiote-aam	replicated	1/1	symbioteh2020/symbiote-aam:5.0.9	*:8080->8080/tcp
k0qpcbcsen4t	symbiote-app_symbiote-cloudconfig	replicated	1/1	symbioteh2020/symbiote-cloudconfig:3.0.1	*:8888->8888/tcp
x1dtr0zq4vhf	symbiote-app_symbiote-eureka	replicated	1/1	symbioteh2020/symbiote-eureka:3.0.1	
xbicya2uastz	symbiote-app_symbiote-mongo	replicated	1/1	mongo:3.6	
rlg6wrafb7q	symbiote-app_symbiote-monitoring	replicated	1/1	symbioteh2020/symbiote-monitoring:3.0.2	
zqfoxpusy9zi	symbiote-app_symbiote-nginx	replicated	1/1	symbioteh2020/symbiote-nginx:3.0.0	*:8102->8102/tcp
urxi50naxqyy	symbiote-app_symbiote-rabbitmq	replicated	1/1	rabbitmq:3-alpine	*:5671-5672->5671-5672/tcp, *:15672->15672/tcp
qyoyx1ktq60j	symbiote-app_symbiote-rap	replicated	1/1	symbioteh2020/symbiote-rap:3.0.4	
my7lnphr3wng	symbiote-app_symbiote-rh	replicated	1/1	symbioteh2020/symbiote-rh:3.0.6	*:8001->8001/tcp

Slika 13: Pokrenuti docker servisi

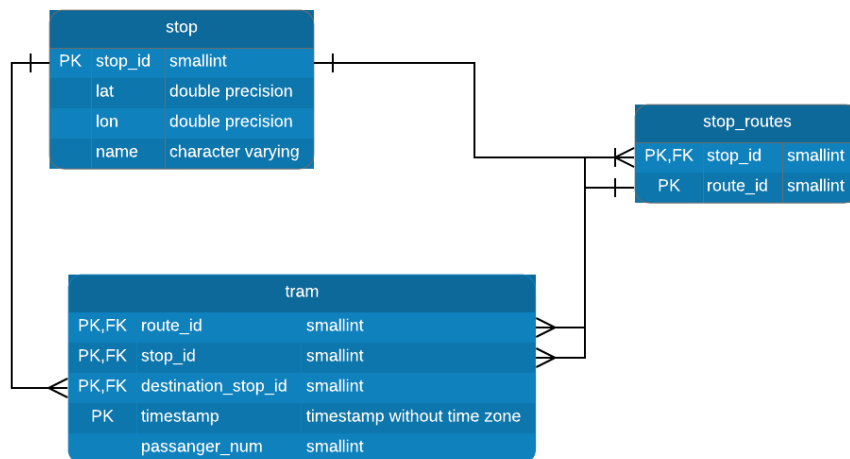
4. Primjena usluge suradnog opažanja okoline i SymbloTe programskog okvira u izradi rješenja za praćenje vozila javnog gradskog prijevoza

4.1. Struktura i pohrana podataka

Baza podataka sastoji se od tri relacijske tablice (slika 14):

- Tramvaj (*tram*)
- Stanica (*stop*)
- Rute stanica (*stop_routes*)

Stop relacijska tablica s atributima *stop_id*, *lat*, *lon* te *name* reprezentira stanicu s pripadajućim jednoznačnim identifikacijskim brojem, njezinu lokaciju, to jest geografsku širinu te duljinu, te ime stanice. *Stop_routes* relacija povezuje stanice i rute tramvaja koje prolaze tom stanicom preko njihovih jedinstvenih identifikacijskih brojeva. *Timestamp* atribut dio je relacije *tram* te predstavlja vremensku oznaku s informacijom kada je vozilo na ruti s oznakom *route_id*, u smjeru odredišne stanice s identifikacijskim brojem *destination_stop_id*, viđeno na stanici s oznakom *stop_id*. Osim četiri navedena atributa, dio relacije *tram* je *passanger_num* koji predstavlja broj putnika u tom vozilu.



Slika 14: Baza podataka

4.2. Registracija resursa na SymbloTe Core

Po završetku konfiguracije i pokretanja komponenti SymbloTe Clouda, omogućena je registracija resursa platforme na SymbloTe Core. Resurse je moguće registrirati na dva načina:

- Registracija korištenjem JSON formata
- Registracija korištenjem JSON reprezentacije RDF modela

Odabran je pristup registracije resursa putem JSON formata u sklopu kojeg je moguće registrirati tri tipa resursa:

- senzor
- aktuator
- servis

Za potrebe ovog rada registrirano je sedam senzora, gdje svaki predstavlja jednu tramvajsku stanicu na Vukovarskoj ulici u Zagrebu, s pripadajućim svojstvima. Polja koja je potrebno popuniti su *internal_id*, odnosno jedinstvena oznaka resursa koji će biti registriran, *plugin_id*, to jest jedinstvena oznaka RAP Plugin-a koji pruža uslugu resursima. Također, nužno je navesti *resource* polje, odnosno opis resursa, te *accessPolicy* i *filteringPolicy* koji predstavljaju razinu privatnosti pristupa i

pretrage pojedinog resursa, a koji su ovom slučaju postavljeni na vrijednost *PUBLIC* te su shodno tome vidljivi svim korisnicima SymbloTe projekta.

Proizvoljno odabrani parametri polja *resource* jednog senzora su unutar grupe *locatedAt* i predstavljaju podatke o lokaciji jednog senzora, odnosno tramvajske stanice. Upravljač registracije koristi sučelje za međudjelovanje kako bi komunicirao sa SymbloTe Core-om te registrirao resurse na određenu platformu. Također, potrebno je navesti URL usluge za međudjelovanje, to jest *ngrok* URL, koji je naveden i prilikom registracije same platforme.

Uz navedene, dodani su i „*observesProperty*“ te „*hasProperty*“ proizvoljni parametri, koji sadrže svojstva senzora koja su predmet promatranja. U konkretnom slučaju to su „*number*“, koji reprezentira broj putnika, te „*timeConstant*“, odnosno vrijeme dolaska tramvaja na određenu stanicu. Primjer JSON formata jednog prijavljenog senzora dan je na Slici 15.

```
[{
  "internalId": "6",
  "pluginId": "RapPluginForYPlatform",
  "accessPolicy": {
    "policyType": "PUBLIC",
    "requiredClaims": {}
  },
  "filteringPolicy": {
    "policyType": "PUBLIC",
    "requiredClaims": {}
  },
  "resource": {
    "@c": ".StationarySensor",
    "name": "Kruge",
    "description": [
      "Tram stop"
    ],
    "featureOfInterest": {
      "name": "Tram number, arrival time and number of passangers",
      "description": [
        "Shows next tram number, time until it arrives in minutes and number of passangers inside it"
      ],
      "hasProperty": [
        "number",
        "timeConstant"
      ]
    },
    "observesProperty": [
      "number",
      "timeConstant"
    ],
    "locatedAt": {
      "@c": ".WGS84Location",
      "longitude": 15.985374,
      "latitude": 45.800547,
      "altitude": 60,
      "name": "Zagreb",
      "description": [
        "Stop sensor"
      ]
    },
    "interworkingServiceURL": "https://ebd8ccb8.ngrok.io"
  }
}]
```

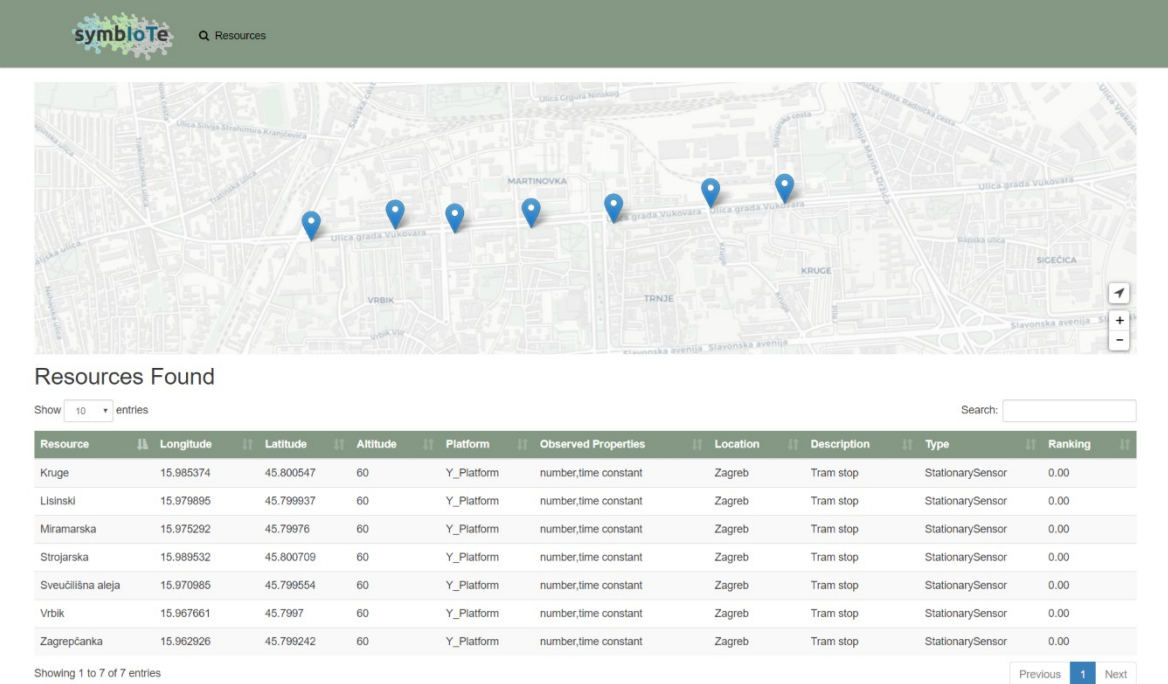
Slika 15: Opis jednog senzora

Sama registracija obavljena je slanjem HTTP POST zahtjeva u kojem se šalje opis svih senzora u JSON datoteci „resources1.json“ na krajnju točku, odnosno URL na kojem je pokrenuta komponenta Cloud-a, *RegistrationHandler* (slika 16).

```
➔ symbiote curl -X POST \
  http://localhost:8001/resources \
  -H 'cache-control: no-cache' \
  -H 'content-type: application/json' \
  -d @resources1.json
```

Slika 16: Registracija resursa na SymbloTe Core

Rezultat registracije je prikaz resursa na SymbloTe grafičkom korisničkom sučelju dostupnom na poveznici pod [10]. Potrebno je unijeti ime platforme uneseno prilikom registracije te odabrati opciju „Search“. Izlaz dobiven prijavom resursa vidljiv je na Slici 17.



Resources Found

Show entries Search:

Resource	Longitude	Latitude	Altitude	Platform	Observed Properties	Location	Description	Type	Ranking
Kruga	15.985374	45.800547	60	Y_Platform	number,time constant	Zagreb	Tram stop	StationarySensor	0.00
Lisinski	15.979895	45.799937	60	Y_Platform	number,time constant	Zagreb	Tram stop	StationarySensor	0.00
Miramarska	15.975292	45.79976	60	Y_Platform	number,time constant	Zagreb	Tram stop	StationarySensor	0.00
Strojarska	15.989532	45.800709	60	Y_Platform	number,time constant	Zagreb	Tram stop	StationarySensor	0.00
Sveučilišna aleja	15.970985	45.799554	60	Y_Platform	number,time constant	Zagreb	Tram stop	StationarySensor	0.00
Vrbik	15.967661	45.7997	60	Y_Platform	number,time constant	Zagreb	Tram stop	StationarySensor	0.00
Zagrepčanka	15.962926	45.799242	60	Y_Platform	number,time constant	Zagreb	Tram stop	StationarySensor	0.00

Showing 1 to 7 of 7 entries Previous **1** Next

Slika 17: Prikaz registriranih resursa

4.3. RAP Plugin

Tri su načina za izradu vlastite implementacije RAP Plugin-a:

- 1) Prilagodba internog RAP Plugin-a
- 2) Izrada koristeći RAP Plugin početnog paketa
- 3) Izrada RAP Plugin-a u drugim jezicima

U ovom potpoglavlju prikazana je izrada RAP Plugin-a korištenjem početnog paketa čija je zamisao da se koristi kao ovisnost u implementaciji koja povezuje aplikaciju sa SymbloTe-om. Implementira generičke dijelove poput RabbitMQ komunikacije s RAP komponentom Cloud-a. Na taj način programer ne mora provoditi složenu komunikaciju [11]. Za izradu je najprije nužno kreirati novi SpringBoot projekt u programskom jeziku Java bez početnih ovisnosti te potom dodati ovisnosti za određenu verziju RAP Plugin početnog paketa, u ovom konkretnom slučaju 1.0.0, u datoteku „build.gradle“. Također, potrebno je dodati ovisnosti potrebne za povezivanje s bazom podataka te jitpack repozitorijem koji je potreban za rad Plugin-a (slika 18).

```
repositories {  
    mavenCentral()  
    maven { url "https://jitpack.io" } // this is important to add  
}  
  
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
    implementation('com.github.symbiote-h2020:ResourceAccessProxyPluginStarter:1.0.0')  
    implementation("org.springframework.boot:spring-boot-starter")  
    implementation("org.springframework.boot:spring-boot-starter-jdbc")  
    implementation("org.postgresql:postgresql")  
}
```

Slika 18: Ovisnosti unutar "build.gradle" datoteke

Unutar datoteke „application.properties“ nužno je dodati konfiguracijske podatke prikazane na Slici 19. Prva linija datoteke sadrži ime Plugin-a koje je navedeno kao „pluginId“ prilikom registracije resursa na SymbloTe Core. Druga skupina parametara odnosi se na konfiguraciju RabbitMQ poslužitelja uz čiju pomoć RAP Plugin komunicira s RAP komponentom SymbloTe Cloud-a, dok se treća skupina

odnosi na konfiguraciju samog Plugin-a. Ukoliko „*filtersSupported*“ ima vrijednost *false*, RAP komponenta filtrira odgovore dobivene od strane Plugin-a. Shodno tome, potrebno je vratiti sva zapažanja senzora, dok ako je postavljena vrijednost *true*, implementacijom Plugin-a se podaci filtriraju. Definirana vrijednost parametra „*notificationsSupported*“ je *false* što označava da strujanje podataka nije moguće [12].

```
spring.application.name=RapPluginForYPlatform

rabbit.host=localhost
rabbit.username=guest
rabbit.password=guest
rabbit.replyTimeout=60000

rap-plugin.filtersSupported=false
rap-plugin.notificationsSupported=false

spring.datasource.url=jdbc:postgresql://oh2.tel.fer.hr:5432/tramcheckDB
spring.datasource.username=postgres
spring.datasource.password=postgres
```

Slika 19: Konfiguracija "application.properties" datoteke

Unutar projekta je stvoreno nekoliko klasa od kojih su najvažnije *DatabaseHelper*, *Stop*, *Tram*, *RapPluginApplication* te *PluginResourceAccessListener*. *DatabaseHelper* klasa sadrži metode za dohvaćanje podataka iz baze. Metode izvršavaju upite u bazu uz pomoć JDBC (engl. *Java Database Conectivity*) programskog sučelja. *Stop* i *Tram* objekti predstavljaju entitete iz baze sa svim pripadajućim atributima kao varijablama klase. Unutar *RapPluginApplication* klase nalazi se glavna metoda koja pokretanjem programa pokreće osluškivanje Plugin-a (slika 20).

```

@SpringBootApplication
public class RapPluginApplication implements CommandLineRunner {

    @Autowired
    RapPlugin rapPlugin;

    @Autowired
    PluginResourceAccessListener listener;

    public static void main(String[] args) {
        SpringApplication.run(RapPluginApplication.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        rapPlugin.registerReadingResourceListener(listener);
    }

}

```

Slika 20: Pokretanje osluškivanja RAP Plugin-a

Moguće je implementirati tri vrste metoda, ovisno o resursu, koje osluškuju tijekom rada programa:

- ResourceAccessListener
- ActuatorAccessListener
- ServiceAccessListener

S obzirom na to da su tramvajske stanice registrirane kao senzori te je potrebno prikazati prikupljeno vrijeme dolazaka tramvaja na određenu stanicu, kao i broj putnika, odabran je *ResourceAccessListener* koji sadrži sljedeće metode koje je potrebno nadjačati:

- getResource - čita zadnje očitavanje senzora
- getResourceHistory – čita povijest očitavanja senzora

Metode kao ulazni parametar primaju listu objekata tipa *ResourceInfo*, a implementirane su unutar klase *PluginResourceAccessListener*. *ResourceInfo* predstavlja opis tramvajske stanice za koju se traže informacije.

Za traženu stanicu se pretražuje baza te se pronalaze svi dojavljeni podaci o tramvajima koji prolaze tom stanicom u intervalu od 15 minuta. Potom se računa potrebno vrijeme dolaska do stajališta te ako tramvaj za koji je dojavljena informacija još nije prošao traženu stanicu stvaraju se dva objekta tipa *ObservationValue*. Jedan objekt predstavlja broja putnika u vozilu, dok drugi predstavlja vrijeme u minutama do dolaska opaženog tramvaja na traženu stanicu. Osim vrijednosti opažanja, *ObservationValue* konstruktori, kao parametre primaju objekt tipa *Property*, koji predstavlja svojstvo koje se opaža, te *UnitOfMeasurement*, odnosno mjernu jedinicu opažene vrijednosti (slika 21).

```
String observationValue1 = String.format("passanger number, line=%s, destination=%s",
    tram.getRouteId(), destinationStop.getStopName());
ObservationValue passangerNumObservationValue = new ObservationValue(
    Integer.toString(tram.getPassangerNumber()),
    new Property(observationValue1, "", Arrays.asList("Number of passangers on specific tram line and with destination stop as direction.")),
    new UnitOfMeasurement("", "", "", null));

ArrayList<ObservationValue> passangerNumberObservationValueList = new ArrayList<>();
passangerNumberObservationValueList.add(passangerNumObservationValue);

Observation passangerNumberObservation = new Observation(Integer.toString(waitingStop), loc, timestamp, samplet,
    passangerNumberObservationValueList);
observationList.add(passangerNumberObservation);
```

Slika 21: Stvaranje „ObservationValue“ objekta za vrijednost opažanja broja putnika u vozilu

Svaki kreirani *ObservationValue* objekt, uz lokaciju opažanja, vrijeme stvaranja zapisa o opažanju te resurs kojem se prosljeđuje taj podatak, predaje se u konstruktor objekata *Observation*. Za svaki pronađeni tramvaj u listu tipa *Observation* dodaju se po dva objekta, jedan za broj putnika te jedan za vrijeme dolaska na stanicu u minutama. Dobivena lista se vraća metodi koja osluškuje, a potom i SymbloTe grafičkom korisničkom sučelju s kojeg je moguće uputiti zahtjev te sukladno tome dobiti odgovor. Primjer je dan Slikom 22 gdje su traženi podaci o dolasku tramvaja na stanicu Kruge te su dobivene informacije o dolasku dva tramvaja, linije 5 i 3, te broj putnika u njima.

Kruga data

Show 10 entries

Search:

Measurement	Observed Property	Unit	Latitude	Longitude	Observation Time
5	arrival time, line=3, destination=Strojarska	min	45.799554	15.970985	2019-06-08, 19:25:43
8	arrival time, line=5, destination=Strojarska	min	45.799242	15.962926	2019-06-08, 19:25:43
10	passanger number, line=3, destination=Strojarska	NA	45.799554	15.970985	2019-06-08, 19:25:43
30	passanger number, line=5, destination=Strojarska	NA	45.799242	15.962926	2019-06-08, 19:25:43

Showing 1 to 4 of 4 entries

Previous 1 Next

passanger number, line=5, destination=Strojarska

Generate Graphical Report

Close

Slika 22: Dobivene informacije o vremenu dolasku tramvaja te njihovom broju putnika na stanicu Kruga

5. Zaključak

U ovom radu izrađena je IoT SymbloTe platforma te programsko rješenje za dohvaćanje, obradu i prijenos podataka o lokaciji te broju putnika unutar ZET tramvaja. Informacije prikupljene tehnikom suradnog opažanja su po završetku obrade prenesene na platformu. Osim u svrhu jednostavnog pristupa podacima korištenje IoT platforme omogućava svojstvo interoperabilnosti. Na platformu su, kao resursi, registrirane tramvajske stanice čijim se pretraživanjem, preko RAP Plugin-a, aktiviraju metode koje osluškuju tijekom rada programa. Ukoliko postoji informacija o tramvajskim vozilima koja dolaze na željenu stanicu, korisniku se vraćaju podaci.

Kako bi ovakvo rješenje zaživilo, potreban je veliki broj korisnika radi točnosti podataka o broju putnika te pokrivenosti infrastrukture korisnicima koji bi dijelili podatke. Također, potrebno je proširiti sustav izradom mobilnih aplikacija sa prilagođenim grafičkim sučeljem putem kojih bi korisnici slali zahtjevi za podacima te primali odgovore s informacijama.

6. Literatura

- [1] McClelland, C., *What is an IoT platform?*, dostupno na <https://www.iotforall.com/what-is-an-iot-platform/>
- [2] Stojić, A., Reputacijski sustavi za mobilne aplikacije temeljene na suradnom opažanju okoline, Diplomski rad, Fakultet elektrotehnike i računarstva, 2014.
- [3] SymbloTe, *SymbloTe vision*, dostupno na <https://www.symbiote-h2020.eu/>
- [4] SymbloTe_Git hub, dostupno na <https://github.com/symbiote-h2020/SymbioteCloud>
- [5] SymbloTe, How to make an iot platform symbiote enabled, <https://www.symbiote-h2020.eu/blog/2017/11/03/how-to-make-an-iot-platform-symbiote-enabled/>
- [6] Ngrok, *Download and setup ngrok*, <https://ngrok.com/download>
- [7] SymbloTe administration, <https://symbiote-open.man.poznan.pl/administration>
- [8] Docker, *Get Docker CE for Ubuntu*, <https://docs.docker.com/v17.12/install/linux/docker-ce/ubuntu/#install-docker-ce>
- [9] SymbloTe Git hub, <https://github.com/symbiote-h2020/SymbioteCloud/tree/master/resources/docker/docker-compose/L1>
- [10] SymbloTe Search, <https://symbiote-open.man.poznan.pl/symbioteSearch/>
- [11] SymbloTe Git hub, *Using RAP Plugin starter*, <https://github.com/symbiote-h2020/SymbioteCloud/wiki/4.2-Using-RAP-plugin-starter>
- [12] SymbloTe Git hub, *ResourceAccessProxy Starter*, <https://github.com/symbiote-h2020/ResourceAccessProxyPluginStarter>

Praćenje vozila javnog gradskog prijevoza

Koncepti pametnih gradova nude razna programska rješenja kojima je cilj poboljšati kvalitetu života. Jedna od mnogih značajki je učinkovitije korištenje prometne infrastrukture, u svrhu čega je izrađen ovaj rad. Podaci o lokaciji vozila javnog prijevoza grada Zagreba, prikupljeni uz pomoć korisnika uslugama suradnog opažanja, obrađuju se na zahtjev u programskom jeziku Java. Rezultat obrade je vrijeme u minutama do dolaska vozila na određenu stanicu te pripadajući broj putnika. Dobiveni podaci su preneseni na SymbloTe platformu, IoT platformu koja je izrađena u sklopu ovog rada, a koja nudi mogućnost suradnje sa drugim IoT platformama. Time je osiguran siguran prijenos podataka te interoperabilnost samog programskog rješenja.

Ključne riječi: usluge suradnog opažanja, SymbloTe, resursi, interoperabilnost, IoT platforma

Tracking of vehicles in public transport

Smart city concepts offer a variety of software solutions aimed at improving the quality of life. One of many characteristics is the more efficient usage of the traffic infrastructure, for the purpose of which this work was developed. Information on the location of public transport vehicle in the city of Zagreb, collected through the mobile crowdsensing technique, is processed on request in the Java programming language. The result is the time in minutes until the vehicle arrives at the wanted station with the corresponding number of passengers. The obtained data is being transferred on SymbloTe platform, IoT platform which was developed for the purposes of this work, and which offers the possibility of cooperation with other IoT platforms. This ensures secure data transfer and interoperability of the software solution.

Keywords: mobile crowdsensing, SymbloTe, resources, interoperability, IoT platform