

# PROJEKAT 2 – IoTS Smart-Building

Cilj ovog projekta je demonstracija implementacije mikroservisne arhitekture u različitim tehnologijama, pružajući studentima priliku da steknu praktično iskustvo u razvoju i integraciji mikroservisa koristeći popularne tehnologije kao što su .NET, NodeJS, Java/Spring Boot, Python/Flask.

Prvi mikroservis, nazvan Sensor Dummy, simulira očitavanje podataka sa senzora i šalje ih na određeni topic MQTT brokera. Drugi mikroservis, nazvan Analytics, pretplaćuje se na MQTT message broker i analizira primljene podatke u cilju detektovanja anomalija ili značajnih događaja. Rezultati analize se skladište u NoSQL bazi podataka, InfluxDB.

U daljem tekstu će biti opisane ključne komponente sistema, njihova funkcionalnost i način njihovog međusobnog povezivanja, kao i koraci za pokretanje i testiranje sistema pomoću Docker kontejnera.

## Sensor Dummy mikroservis

SensorDummy servis je implementiran u programskom jeziku Javascript i koristi Node.js okruženje. Koristi dodatne npm pakete za čitanje podataka iz CSV fajla i za komunikaciju sa MQTT brokerom. Prilikom pisanja Docker Compose fajla, važno je voditi računa da se kontejner za SensorDummy servis pokrene tek nakon što se EMQX broker startuje. SensorDummy servis čita podatke iz CSV fajla data.csv i objavljuje ih na topicu "sensor\_dummy/values" MQTT brokera.

Kao MQTT broker izabran je EMQX broker.

## Analytics mikroservis

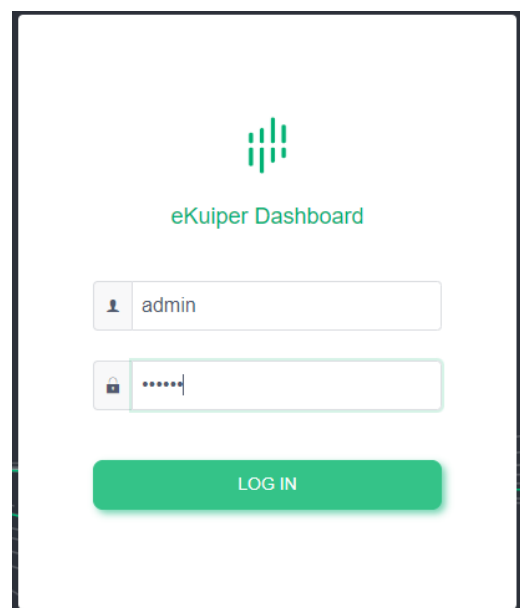
Uloga ovog mikroservisa jeste da čita podatke sa sensor\_dummy/values topica odgovarajućeg MQTT brokera i pročitane podatke upisuje na analytics/values topic, preko koga se pročitani podaci prosleđuju eKuiper-u.

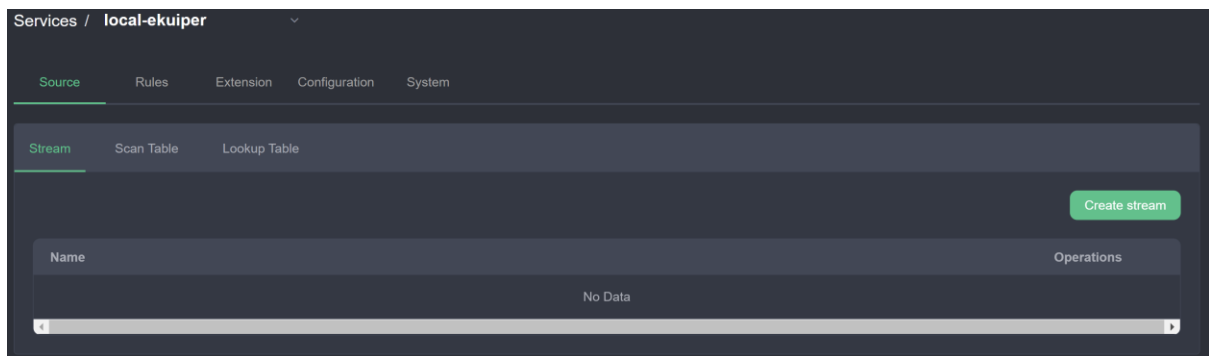
Analytics servis je napisan u .NET-u i koristi odgovarajuće biblioteke za rad sa MQTT-om i sa InfluxDB bazom podataka. Za ovaj mikroservis takođe važi da treba biti pokrenut nakon što je MQTT broker startovan.

## Podešavanje eKuiper-a

Odgovarajuću konfiguraciju eKuiper-a, pokrenutog u okviru kontejnera unosimo preko GUI-ja koji nudi management konzola

1. Pre svega neophodno je ulogovati se korišćenjem sledećih parametara:
  - Username: admin
  - Password: public
  -
2. Selektujemo local-ekuiper servis i biramo opciju Create Stream u okviru Source > Stream kartice





3. Unosimo sledeće parametre u okviru Create Stream dijaloga:

- Stream Name: `iot2stream`
- Stream Type: `mqtt`
- Data Source: `analytics/values`

Na kraju biramo opciju Submit.

The screenshot shows the 'Create Stream' dialog box. It has a dark theme. Fields include: 'Stream Name' with value 'iot2stream', 'Whether the schema stream' with an unchecked checkbox, 'Stream Type' dropdown with 'mqtt', 'Data Source (MQTT Topic)' with value 'analytics/values', 'Configuration key' dropdown with 'Select', 'Stream Format' dropdown with 'json', and 'Shared' with radio buttons for 'true' and 'false' (the 'false' option is selected). At the bottom are 'Submit' and 'Cancel' buttons.

4. Dalje je potrebno uneti pravilo po kome će se filtrirati podaci koji pristižu. U okviru Rules kartice, biramo Create Rule opciju. Unosimo jedinstveni id pravila i odgovarajući opis, a potom unosimo SQL upit kojim se selektuju podaci iz skupa svih podataka koji pristižu, po odgovarajućem kriterijumu. U našem slučaju, selektovaćemo sva merenja kod kojih je koncentracija ugljen-dioksida veća od 495.

\* Rule ID

eKuiperRule

Name

The rule that triggers an alarm when the CO2 contentration is too high

\* SQL

```

1 SELECT *
2 FROM iot2stream
3 WHERE co2 > 495

```

5. Pored ovoga, želimo da unesemo i odgovarajuću akciju koja će se izvršiti kada naiđe podatak koji zadovoljava navedeno pravilo. Selektujemo opciju Add u okviru Actions dela. Parametri koje unosimo su sledeći:

- Sink: mqtt
- MQTT broker address: tcp://172.28.48.1:1883 (unosimo ip adresu na kojoj su pokrenuti svi docker kontejneri, može se videti pozivom ipconfig komande)
- MQTT topic: eKuiper/carbonalarm
- Ostale vrednosti ne menjamo

Biramo opciju Test Connection, kako bismo proverili da li je konekcija uspešna, a zatim biramo Submit. I konačno, ponovo biramo Submit i na taj način kreiramo pravilo.

Add action

\* Sink [Documentation](#)

mqtt

Resource ID

+ Add sink template

Select

\* MQTT broker address ?

tcp://172.28.48.1:1883

\* MQTT topic ?

eKuiper/carbonalarm

## Konfiguracija InfluxDB baze podataka

Za smeštanje izmerenih vrednosti kod kojih je koncentracija CO2 veća od dozvoljene koristi se InfluxDB baza podataka. U ovu svrhu, neophodno je na odgovarajući način konfigurisati bazu.

Prilikom inicijalog pokretanja InfluxDB baze, pojaviće se Get Started dijalog. Parametri koje je potrebno uneti su:

- Username: admin
- Password: adminadmin
- Initial Bucket: iot2
- Organization: organization

InfluxDB baza podataka je na ovaj način konfigurisana i upisani podaci se mogu pogledati u okviru Load Data > Buckets kartice:

The screenshot displays the InfluxDB Data Explorer interface. On the left sidebar, navigation options include Home, Load Data, Data Explorer (selected), Notebooks, Dashboards, Tasks, Alerts, and Settings. The main panel shows a table of sensor data with columns: table, measurement, \_field, \_value, \_start, \_stop, \_time, and timestamp. The table contains 12 rows of data for a sensor named 'sensor' at location 'co2'. Below the table, there are pagination controls showing 1 through 5 pages. At the bottom, the Query Editor is visible, showing a query for 'sensor' with filters for '\_measurement' and '\_field', and a group by clause.

table	measurement	_field	_value	_start	_stop	_time	timestamp
id	string	string	float	datetime RFC3339	datetime RFC3339	datetime RFC3339	
8	sensor	co2	529	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:50.000Z	1377982639
1	sensor	co2	535	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:50.000Z	1377982644
2	sensor	co2	535	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:50.000Z	1377982649
3	sensor	co2	527	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:50.000Z	1377982654
4	sensor	co2	535	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:50.000Z	1377982659
5	sensor	co2	542	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:50.000Z	1377982664
6	sensor	co2	535	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:50.000Z	1377982669
7	sensor	co2	533	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:49.000Z	1377982674
8	sensor	co2	528	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:49.000Z	1377982679
9	sensor	co2	528	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:49.000Z	1377982684
10	sensor	co2	536	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:49.000Z	1377982689
11	sensor	co2	527	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:50.000Z	1377982694
12	sensor	co2	543	2023-07-12T20:37:52.075Z	2023-07-12T21:37:52.075Z	2023-07-12T21:37:50.000Z	1377982699

## Arhitektura sistema

