Projekat 3 – Uputstvo

EdgeX

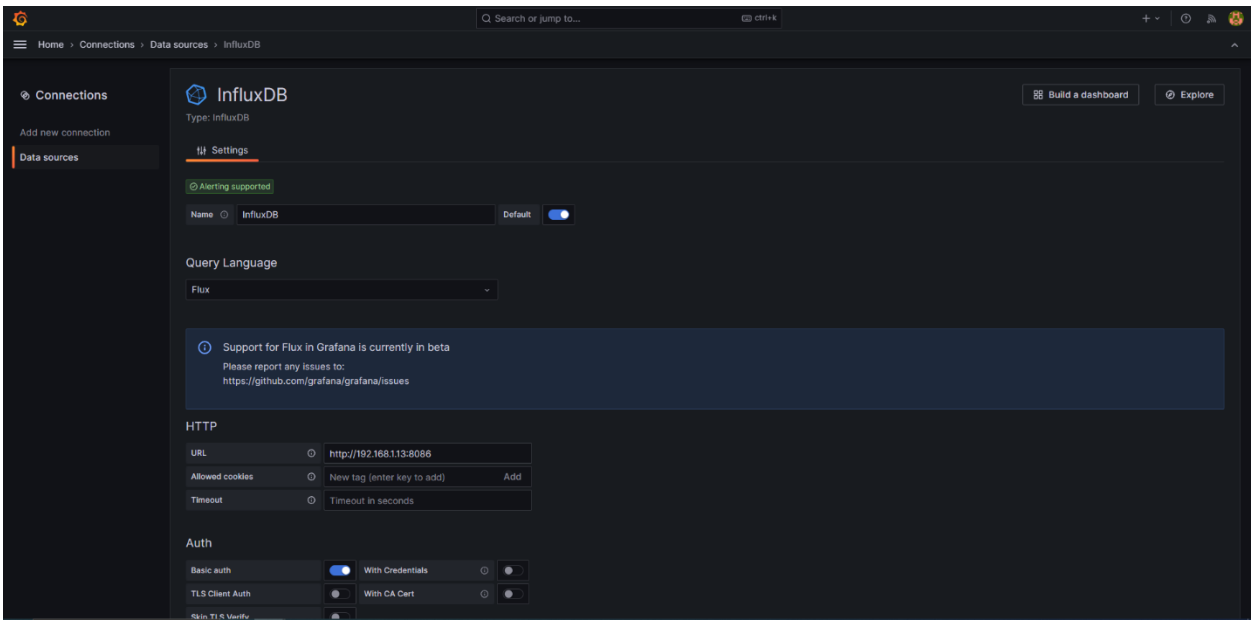
Za postavljanje EdgeX-a potrebno je da se konfiguriše izgled podataka koji dolaze, Device Profile, i Device. U dokumentu *Uputstvo.txt* su navedene sve HTTP metode koje treba da se pozovu sa odgovarajućim sadržajem. Što se tiče MQTT borkera, koristio sam javno dostupni **broker.hivemq.com** na portu **1883**. Topic na koji EdgeX šalje sve podatke koji mu stignu je **edgex/sensor_value**. Skripta *script.py* iz *data-rader-service* foldera se koristi za simulaciju uredjaja koji šalje podatke EdgeX-u.

Visualization service

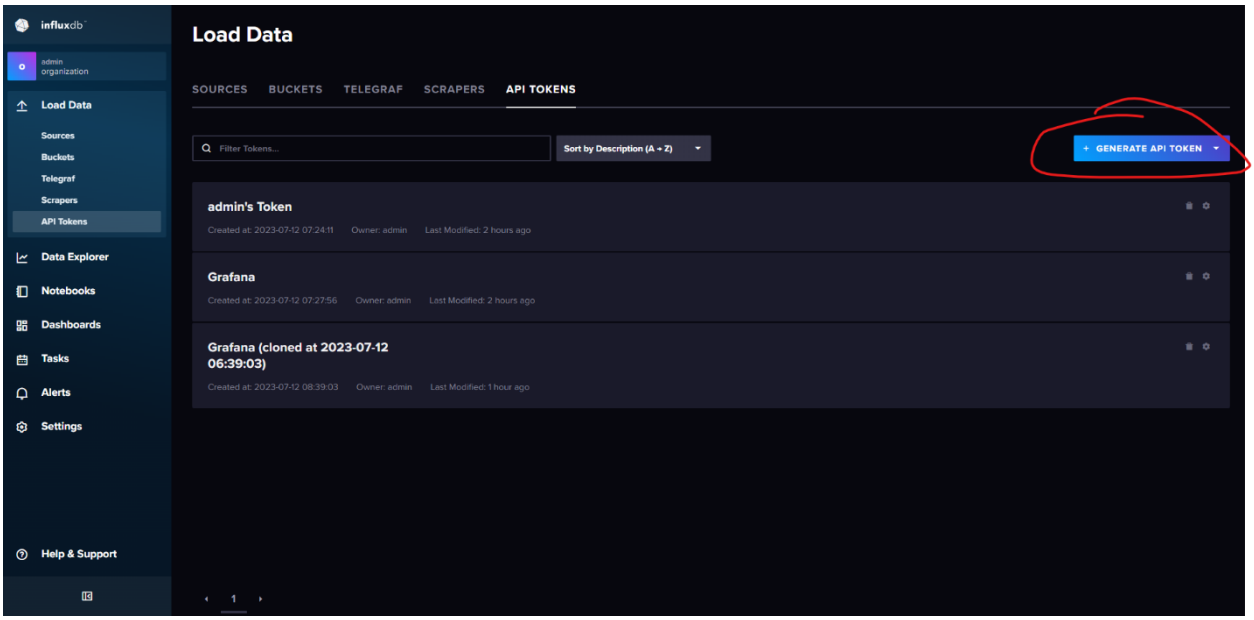
Visualization service je pisan u .Net-u. Cilj mu je da prikupi podatke sa MQTT brokera, i da ih smesti u InfluxDB, gde će se korišćenjem Grafane vizualizovati podaci.

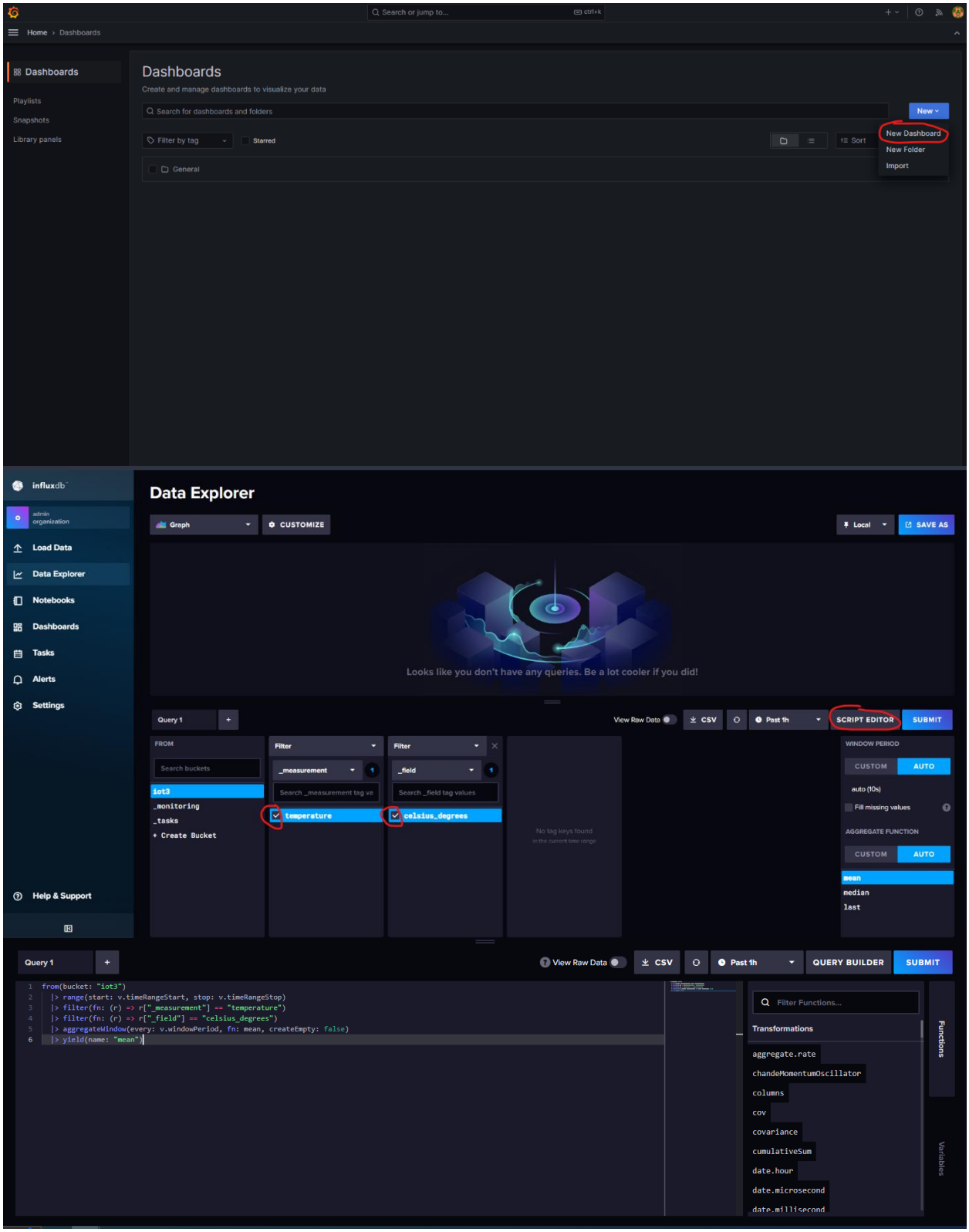
InfluxDb i Grafana

Pri kreiranju naloga na InfluxDB-u na portu 8086, potrebno je uneti username **admin**, password **adminadmin**, organizaciju **organization** i bukcet **iot3**. Nakon toga potrebno je postaviti grafanu. Inicijalni username i password za grafanu je admin, ali kasnije zahteva da se izmeni. Kod Grafane, potrebno je napraviti konekciju sa InfluxDB kao sa slike. Kod URL-a je potrebno staviti

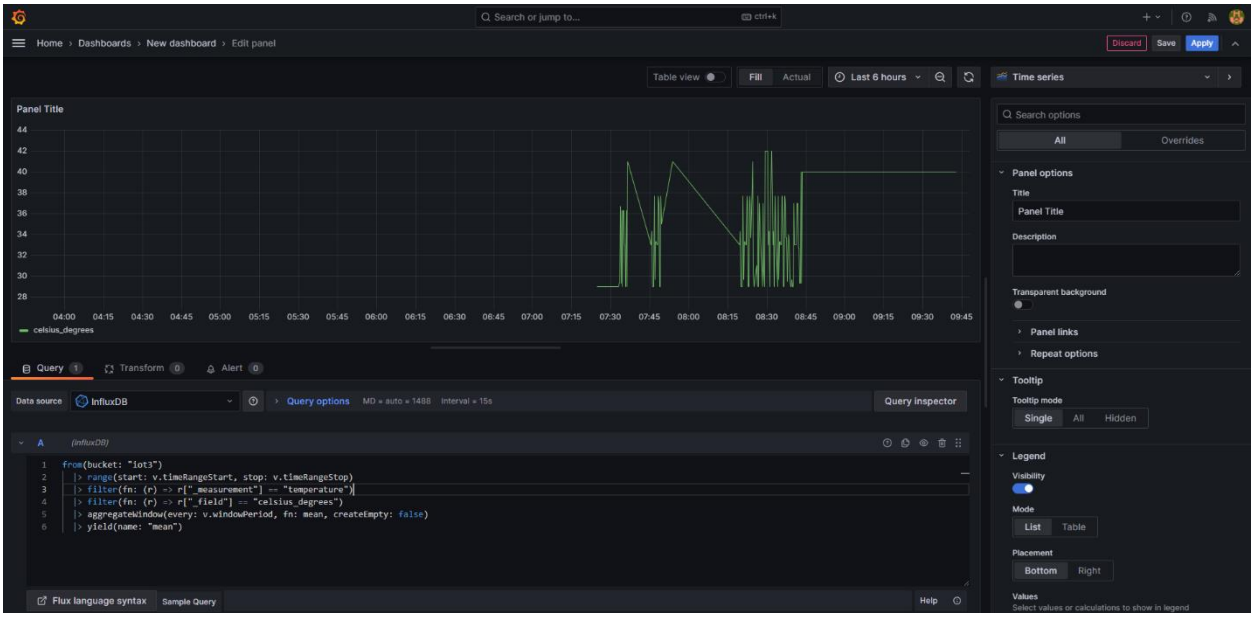


influxdb, jer su i Grafana i InfluxDB pokrenuti u Compose-u i oslanjaju se na istu mrežu, tako da se kontejneri međusobno vide po svom imenu. Da bi se dobio token koji je potreban kod InfluxDB Details dela, potrebno je otići u InfluxDB i generisati novi All Access Token. Zatim je potrebno napraviti **Dashboard** koji će vizualizovati podatke iz konekcije koju selektujemo.
Nakon toga, potrebno je otići u InfluxDB i selektovati podatke koji želimo da vizualizujemo, i kopirati skriptu u Grafani.




Search or jump to...
Home › Dashboards
Dashboards
Playlists
Snapshots
Library panels
Dashboards
Create and manage dashboards to visualize your data
Search for dashboards and folders
New
New Dashboard
New Folder
Import
Filter by tag
Starred
Sort
General
influxdb
admin
organization
Load Data
Data Explorer
Notebooks
Dashboards
Tasks
Alerts
Settings
Help & Support
Data Explorer
Graph
CUSTOMIZE
Local
SAVE AS
Looks like you don't have any queries. Be a lot cooler if you did!
Query 1
View Raw Data
CSV
Past 1h
SCRIPT EDITOR
SUBMIT
FROM
Search buckets
iot3
_monitoring
_tasks
+ Create Bucket
Filter
_measurement
Search _measurement tag va
temperature
Filter
_field
Search _field tag values
celsius_degrees
No tag keys found
in the current time range
WINDOW PERIOD
CUSTOM
AUTO
auto (10s)
Fill missing values
AGGREGATE FUNCTION
CUSTOM
AUTO
mean
median
last
Query 1
View Raw Data
CSV
Past 1h
QUERY BUILDER
SUBMIT
from(bucket: "iot3")
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] == "temperature")
|> filter(fn: (r) => r["_field"] == "celsius_degrees")
|> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
|> yield(name: "mean")
Filter Functions...
Transformations
aggregate.rate
chandeMomentumOscillator
columns
cov
covariance
cumulativeSum
date.hour
date.microsecond
date.millisecond
Functions
Variables

Nakon toga treba da se vidi ovako nešto na ekranu.



Monitoring service

Monitoring service je pisan u node.js-u. Ima za cilj da skupi podatke koje EdgeX šalje na MQTT topic, da detektuje situaciju ukoliko je temperatura van opsega, i da obavesti Device preko EdgeX-a o potrebnim akcijama koje treba da preduzme.
Konfiguracija EdgeX-a kako bi imao svest o eksternom uređaju kojem je potrebno da prosledi podatke:
1. Potrebno je klonirati repozitorijum https://github.com/jonas-werner/colorChanger.git u root folder projekta.
2. Nakon toga je potrebno pozicionirati se u colorChanger folder i izbildovati docker kontejner komandom "docker build -t colorchanger .".
3. Zatim je potrebno pokrenuti ga komandom "docker run -d -p 5000:5000 --name colorchanger colorchanger:latest".
4. Nakon toga je moguce pozivanje REST Api-ja na portu 5000 pomocu Postman-a na adresu http://localhost:5000/api/v1/device/edgexTutorial/changeColor.
5. Sada se komande šalju na uređaj direktno, ali je potrebno da EdgeX bude svestan postojanja tog uređaja, kako bismo preko EdgeX-a mogli da šaljemo komande. To se radi tako sto se opet klonira repozitorijum https://github.com/jonas-werner/EdgeX_Tutorial/tree/master/deviceCreation u root folder projekta. Nakon toga se pozicioniramo u folder **deviceCreation** i pokrenemo skriptu **createRESTDevice.py** sa parametrima localhost za *ip* i WSL IP adresa za *devip*, koja se može videti iz komande **ipconfig**.
6. Pozivom HTTP GET metode sa adrese http://localhost:48082/api/v1/device moramo naći

device TestApp koji smo malopre kreirali pokretanjem Python skripte. Tu se nalazi commands atribut, gde se nalazi PUT, odakle je potrebno preuzeti url adresu. Ta url adresa se koristi za slanje komandi EdgeX-u, koji će kasnije tu komandu proslediti odgovarajućem uređaju(u mom slučaju u index.js fajlu foldera monitoring-service).

Python skripta

Python skriptu je potrebno pokrenuti nakon pokretanja celog sistema. Sistem se pokreće pozivom **docker compose up –build** iz **root** foldera projekta, a skripta se pokreće pozivom **py script.py** iz **data-reader-service** foldera.

Arhitektura projekta