



# Prezentacija projekata

Uroš Stojković 17472

Luka Mladenović 17295



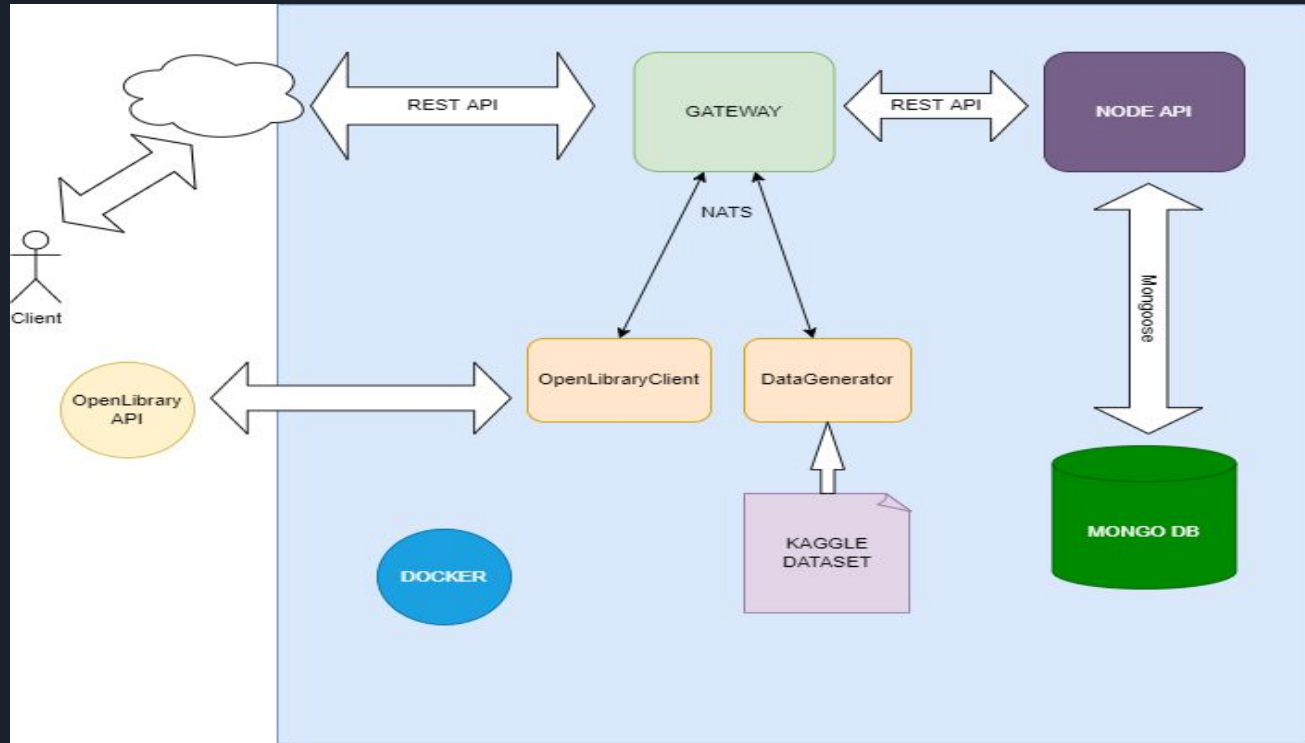
# Projekat 1

OpenLibrary API je korišćen kao public API u ovom projektu

- Podaci su dostupni u JSON formatu
- Omogućava pretragu po naslovu, autoru, žanru, poklapanje teksta, pretragu po specifičnim ID koje biblioteke standardno koriste, izdavačima i sl.

Za dataset je korišćen skup skinut sa Kaggle platforme, sa oko 200-ak knjiga koje igraju ulogu fizičkih kopija knjiga koje se nalaze u biblioteci

# Projekat 1 Arhitektura





# Projekat 1 Detalji

Docker ikonica ukazuje da su svi servisi pokretani u okviru sopstvenog kontejnera preko docker compose komande.

OpenLibrary i Gateway servisi su pisani u ASP.NET, broker između njih je ostao defaultni tj. NATS.

DataGenerator je pisan kao Nodejs skripta

InternalAPI je pisani kao Nodejs skripta, korišćenjem dodatne biblioteke mongoose za laku komunikaciju sa MongoDB bazom podataka.



# Projekat 1 Opis rada

Klijent upućuje zahtev preko REST-a Gateway-u za datom knjigom, korišćenjem imena autora ili naslova tj. dela reči u naslovu knjige.

Vraćanje ili dodavanje nove knjige, kao i rezervisanje knjige iz biblioteke vrši se preko zahteva Gateway-u, koji preko REST-a prosledi HTTP zahtev InternalAPI-ju koji vrši upis odnosno ažuriranje baze podataka. Po završetku operacije, klijent dobija odgovor od Gateway koji zapravo prosleđuje odgovor od InternalAPI koji je on dobio od MongoDB.

DataGenerator je pročitao podatke iz dataset i poslao ih na Gateway endpoint, koji je prosledio podatke InternalAPI-ju koji je izvršio upis u MongoDB.

OpenLibrary servis se obraća public API-ju preko REST-a i pretražuje javnu biblioteku i/ili internet za dodatne informacije o knjizi koju korisnik zahteva.

U slučaju da korisnik nije uspeo da nađe željenu knjigu u MongoDB tj. lokalno u biblioteci, može dobiti dodatne informacije preko public API-ja tako što Gateway servis kombinuje podatke dobijene lokalno tj. iz MongoDB-a i one sa publicAPI-ja kako bi korisniku dao što bolji odgovor.

# Izgled upita

SOA-Gateway / GetByTitle

GET ▼ http://localhost:5001/api/Books/searchTitle?title=potter

Params ● Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON ▼ ≡

```
1 {
2   "inLibrary": [
3     {
4       "ISBN": "0-7475-3269-9",
5       "bookTitle": "Harry Potter and the Philosopher's Stone",
6       "bookAuthor": "J.K. Rowling",
7       "yearOfPublication": "1997",
8       "publisher": "Bloomsbury",
9       "quantity": 3
10    }
11  ],
12  "onInternet": [
13    {
14      "title": "Harry Potter and the Philosopher's Stone",
15      "authors": [
16        "J. K. Rowling"
17      ],
18      "status": "borrow_available",
19      "uri": "https://archive.org/details/harrypotterphilo0000rowl_j1k7"
20    },
21    {
22      "title": "Harry Potter and the Chamber of Secrets",
23      "authors": [
```



# Projekat 2

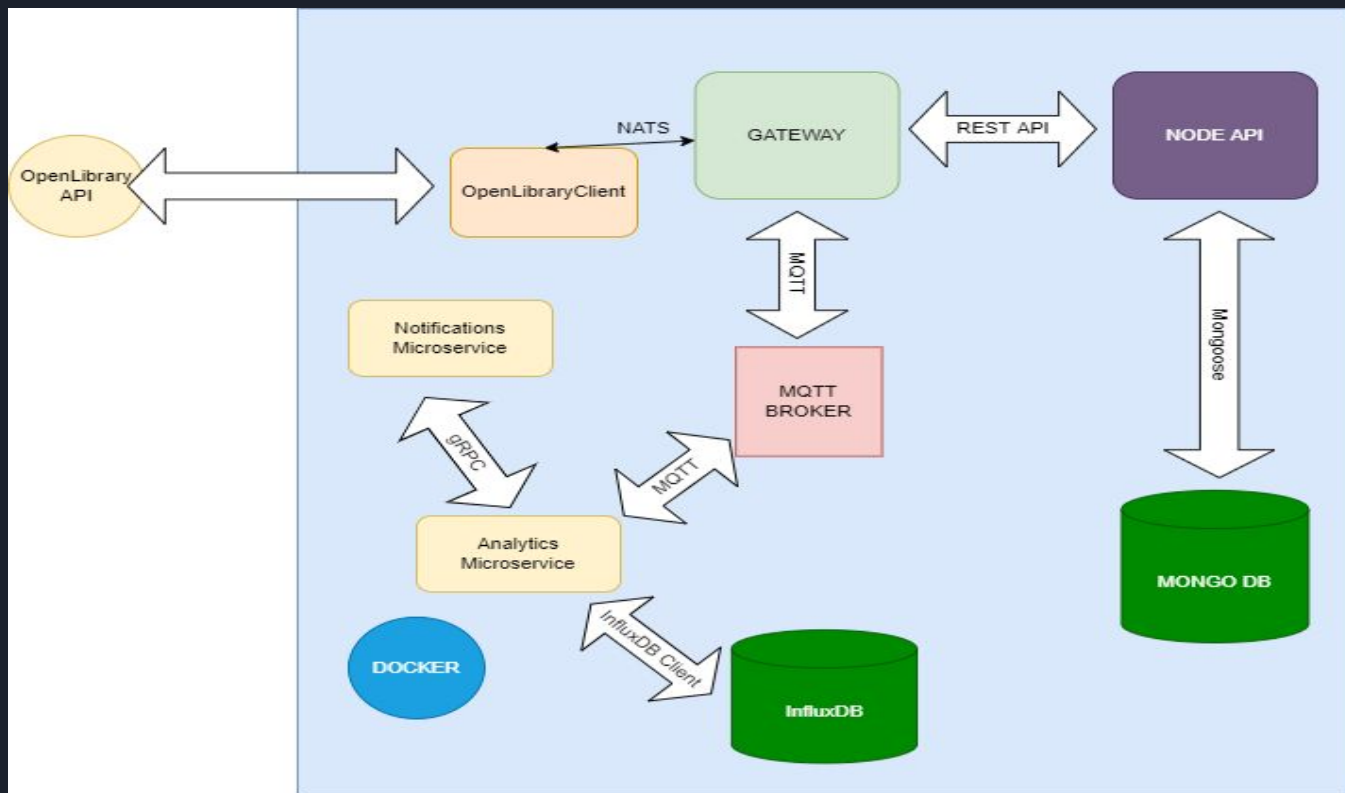
Ovde je uveden Mosquitto MQTT broker na koji Gateway vrši publish-ing.

Podaci koje se publishuju na MQTT protokol izgledaju identično kao i u prvom projektu, šalju se knjige sa istim podacima.

Pravila koja su uvedena prate koliko knjiga je tokom pretraga nađeno u lokalnoj biblioteci a koliko na online biblioteci.

InfluxDB prati ovu metriku koju smo uveli i ako je broj zahteva u prozoru vremena koji pratimo veći od predefinisane vrednosti, to bi značilo da je mikroservis zatrpan zahtevima i da treba kreirati još jedan. Tada AnalyticsMicroservice šalje gRPCa alert servisu za Notifications.

# Projekat 2 Arhitektura





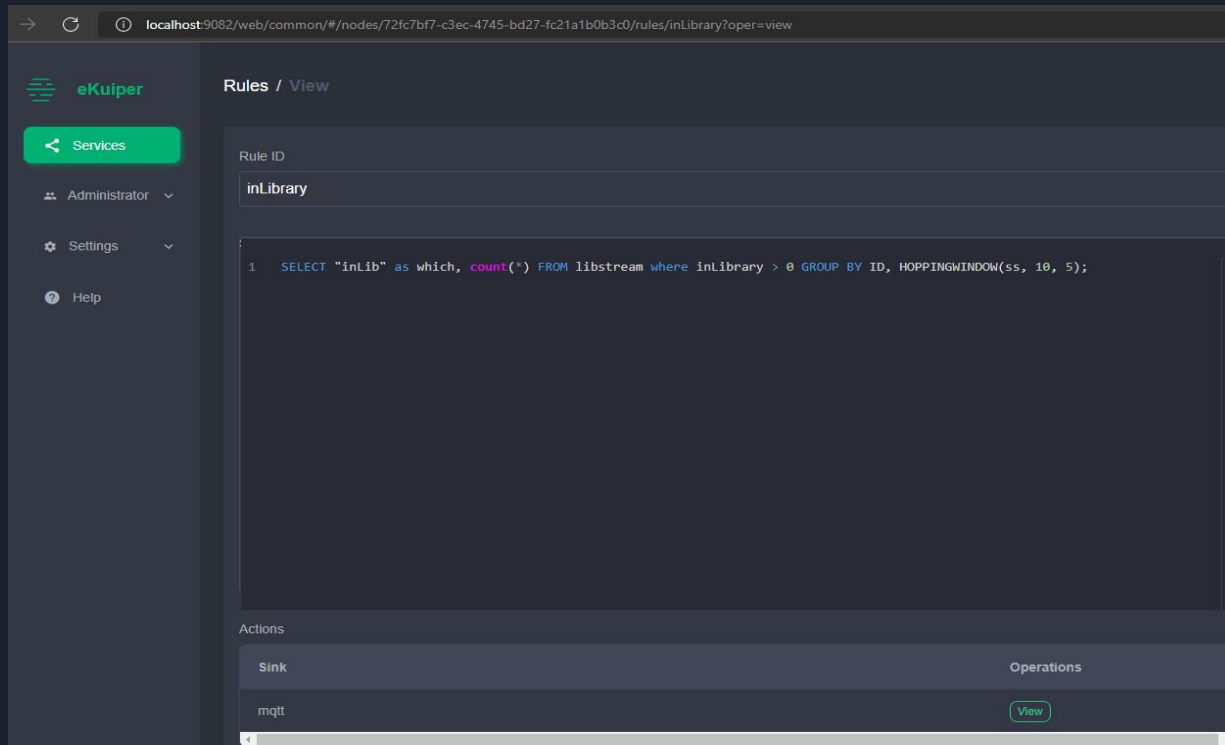


# gRPC format

```
syntax = "proto3";  
  
package Library;  
  
service LibraryService {  
    rpc listBookQueries(bookQuery) returns (statusMessage) {}  
}  
  
message bookQuery {  
    string info = 1;  
}  
  
message statusMessage {  
    string resultInfo = 1;  
}
```

Klijent je implementiran u Nodejs, dok je server implementiran u ASP.NET-u

# Ekuiper izgled pravila



The screenshot displays the eKuiper web interface in a browser window. The address bar shows the URL: `localhost:9082/web/common/#/nodes/72fc7bf7-c3ec-4745-bd27-fc21a1b0b3c0/rules/inLibrary?oper=view`.

**Left Sidebar:**

- eKuiper logo
- Services (highlighted with a green button)
- Administrator (with a dropdown arrow)
- Settings (with a dropdown arrow)
- Help (with a question mark icon)

**Main Content Area:**

Rules / View

Rule ID: inLibrary

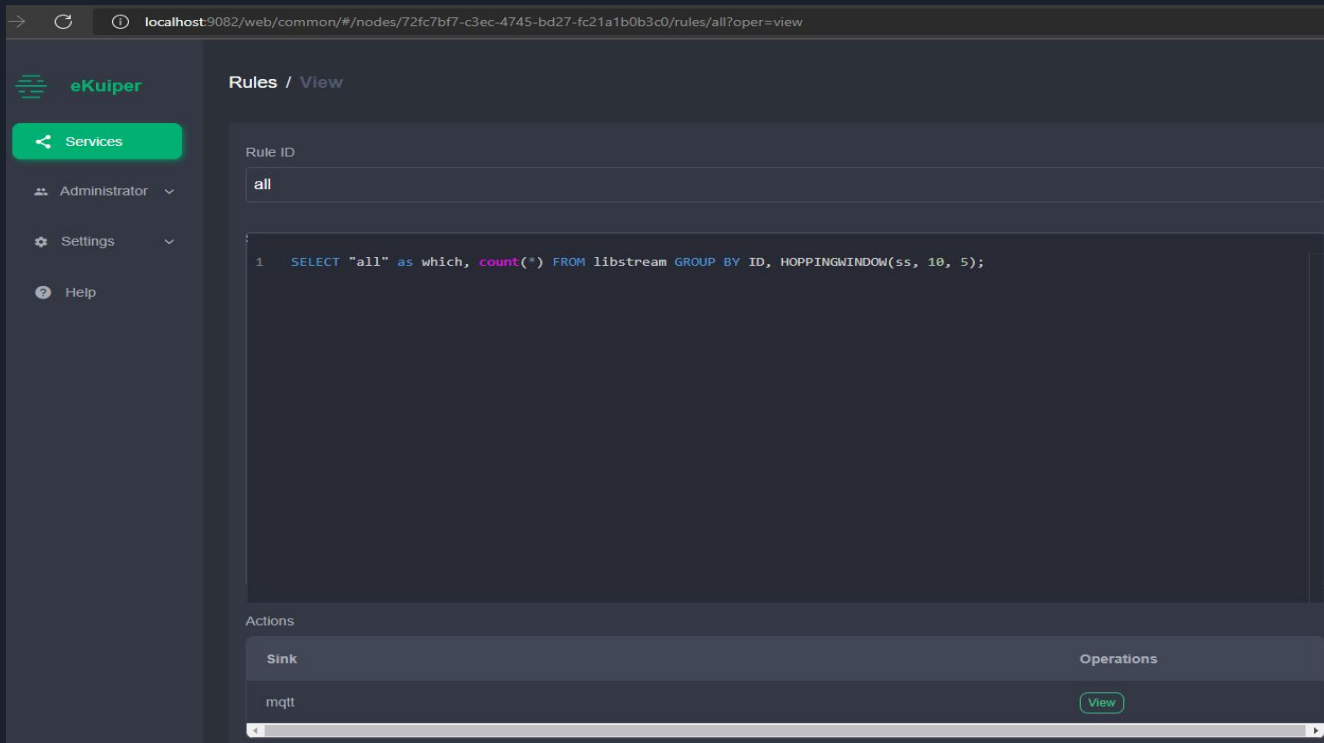
Rule Definition:

```
1 SELECT "inLib" as which, count(*) FROM libstream where inLibrary > 0 GROUP BY ID, HOPPINGWINDOW(ss, 10, 5);
```

**Actions Section:**

Sink	Operations
mqtt	<a href="#">View</a>

# Ekuiper izgled pravila



The screenshot displays the eKuiper web interface in a browser window. The address bar shows the URL: `localhost:9082/web/common/#/nodes/72fc7bf7-c3ec-4745-bd27-fc21a1b0b3c0/rules/all?oper=view`.

The interface has a dark theme. On the left is a sidebar with the eKuiper logo and a menu containing: **Services** (highlighted in green), **Administrator**, **Settings**, and **Help**.

The main content area is titled **Rules / View**. It features a **Rule ID** field containing the text `all`. Below this is a large text area for the rule definition, which contains the following SQL query:

```
1 SELECT "all" as which, count(*) FROM libstream GROUP BY ID, HOPPINGWINDOW(ss, 10, 5);
```

At the bottom of the interface is an **Actions** section. It contains a table with two columns: **Sink** and **Operations**. The **Sink** column has the value `mqtt`. The **Operations** column has a **View** button.

Sink	Operations
mqtt	<a href="#">View</a>

# Ekuiper akcija za pravila

The screenshot displays the eKuiper web interface at `localhost:9082/web/common/#/nodes/72fc7bf7-c3ec-4745-bd27-fc21a1b0b3c0/rules/all?oper=view`. The interface is divided into a left sidebar, a central rule editor, and a right-hand configuration panel.

**Left Sidebar:** Contains the eKuiper logo, a 'Services' button, and a menu with 'Administrator', 'Settings', and 'Help'.

**Central Rule Editor:** Shows a rule named 'all' with the following SQL query:

```
1 SELECT "all" as which, count(*) FROM libstream GROUP BY ID, HOPPINGWINDOW(ss, 10, 5);
```

Below the query, the 'Actions' section lists 'Sink' and 'mqtt', with 'mqtt' selected.

**Right-Hand Configuration Panel (View action):** This panel is titled 'View action' and contains the following settings:

- \* Sink:** A dropdown menu showing 'mqtt'.
- Connection selector:** A dropdown menu.
- MQTT broker address:** A text field containing 'tcp://mosquitto:1883'.
- MQTT topic:** A text field containing 'soa/analytics'.
- MQTT ClientID:** A text field.
- MQTT protocol version:** A dropdown menu.
- QoS:** A dropdown menu.
- Username:** A text field.
- Password:** A text field with a toggle icon.
- Certification path:** A text field.
- Private key path:** A text field.
- Root Ca path:** A text field.
- Skip Certification verification:** Radio buttons for 'True' and 'False'.
- Omit if content is empty:** Radio buttons for 'True' and 'False'.
- Send single:** Radio buttons for 'True' and 'False'.
- Data template:** A text field.

At the bottom of the interface, there are links for 'GitHub' and 'admin'.



# Projekat 3 Opis

Dataset koji se koristi u trećem projektu se sastoji iz četiri skupa podataka, tačnije podaci o potrošnji električne energije vezane za nekoliko priključaka tj. mašine za pranje sudova, peći, frižidera i manjih električnih uređaja koji se mogu naći u kancelariji.

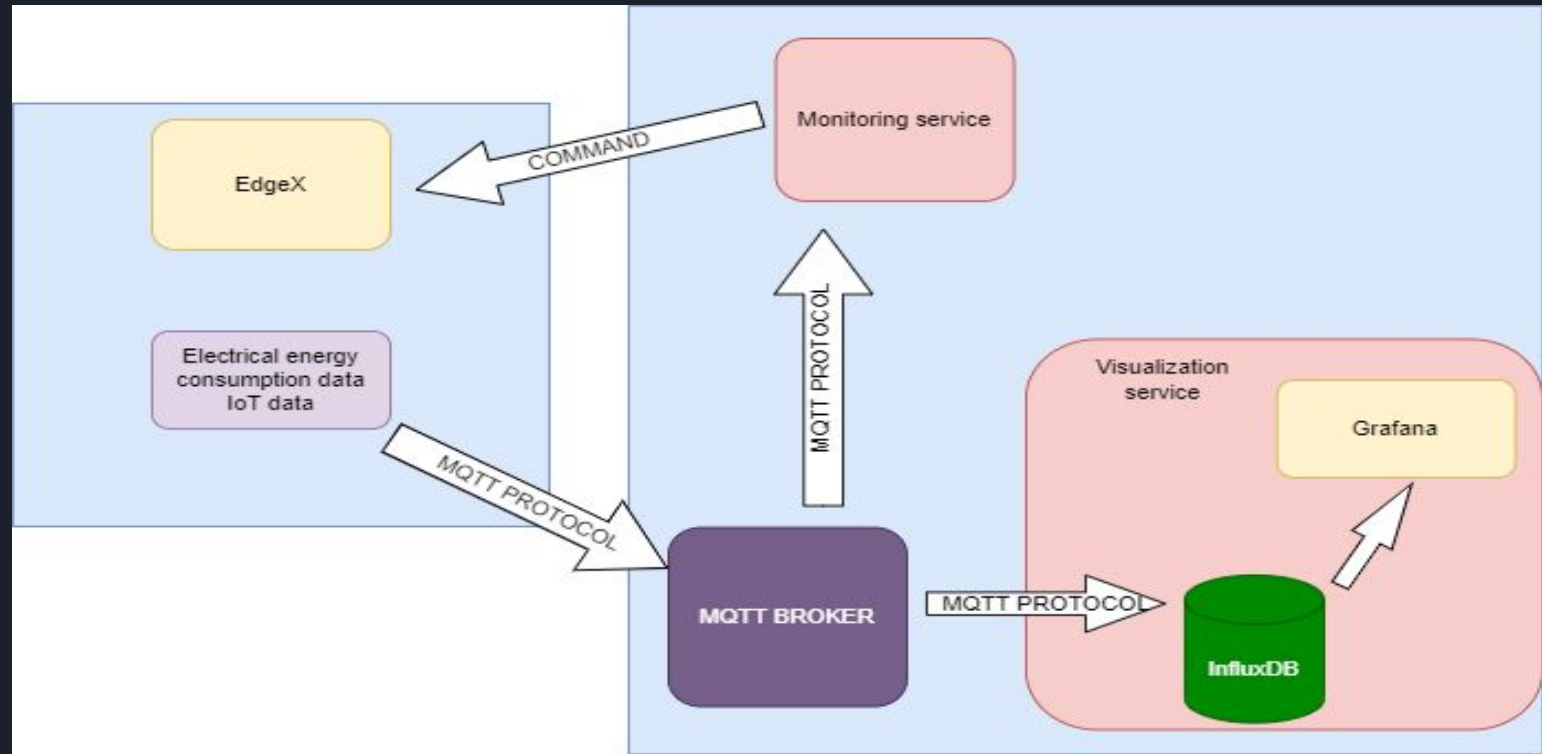
Visualization service je taj koji upisuje podatke u InfluxDB korišćenjem MQTT protokola. Podaci koji se ove pominju su praćeni korišćenjem Grafane.

Monitoring service prati ove podatke i za svaki od priključaka računa trenutno uprosečenu vrednost. Ako zbir svih prosečnih vrednosti pređe određeni prag, servis šalje komandu EdgeX-u da je potrebno uključiti power saving mode.

EdgeX primi ovaj zahtev i prosleđuje komandu pomoćnoj aplikaciji preko REST API-ja. Ona je zadužena samo da loguje tu promenu stanja u konzoli.

Suprotno, kada potrošnja ukupne električne energije padne ispod određenog praga, monitoring servis će poslati zahtev za komandom da je potrebno isključiti power saving mode. Tok podataka za ovaj slučaj je identičan prethodno objašnjenom.

# Projekat 3 Arhitektura



# Projekat 3 Device

GET http://localhost:48082/api/v1/device ...

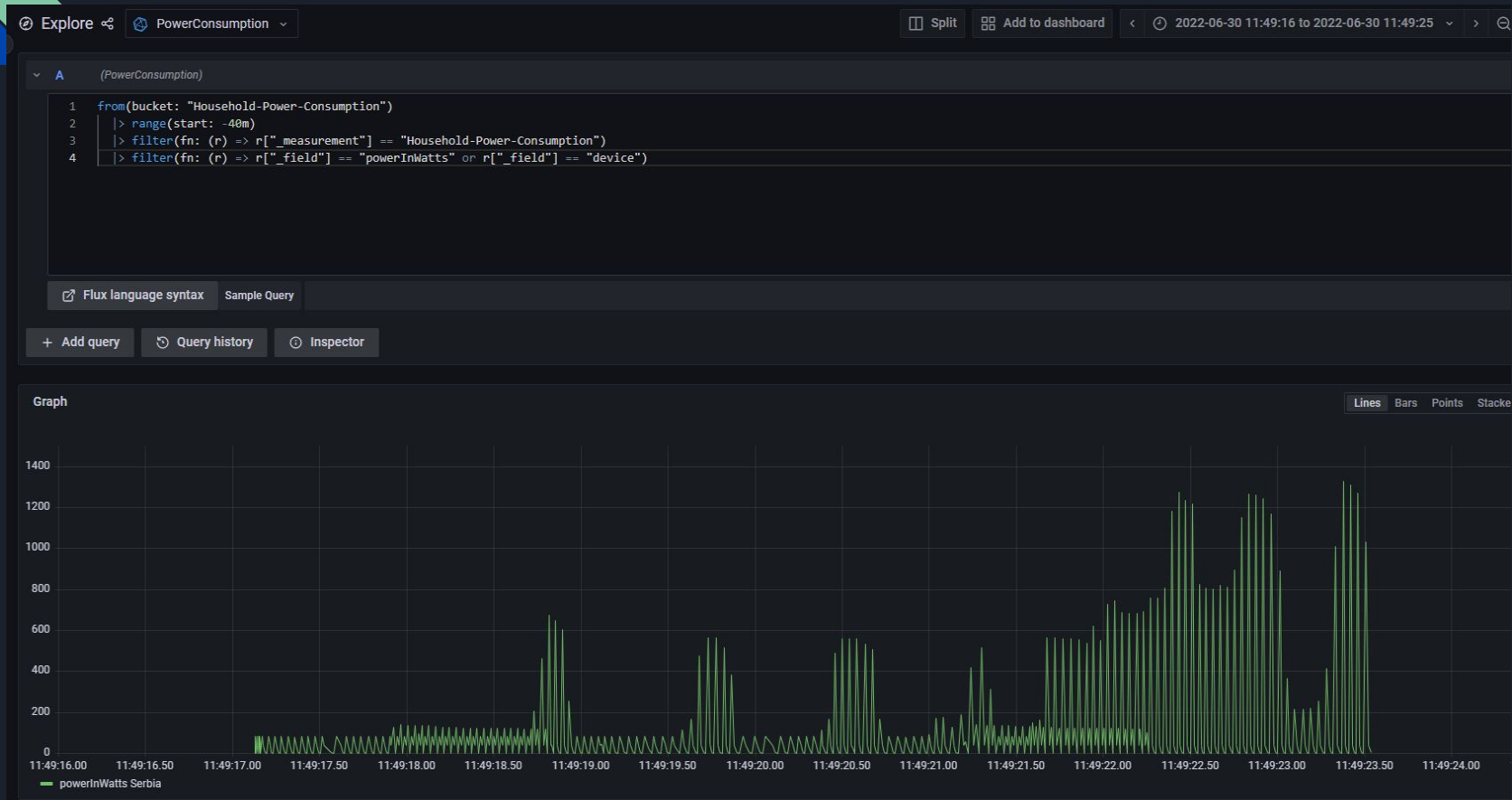
Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (3) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
32      "commands": [
33        {
34          "created": 1655682323941,
35          "modified": 1655682323941,
36          "id": "27da2d31-d4a0-44a3-add0-4e1abe5b7309",
37          "name": "savingMode",
38          "get": {
39            "path": "/api/device/savingMode",
40            "responses": [
41              {
42                "code": "200",
43                "description": "get current savingMode",
44                "expectedValues": [
45                  "savingMode"
46                ]
47              },
48              {
49                "code": "503",
50                "description": "service unavailable"
51              }
52            ],
53            "url": "http://edgex-core-command:48082/api/v1/device/20558a1e-9c51-4ad0-be63-3fc100fed8dc/command/27da2d31-d4a0-44a3-add0-4e1abe5b7309"
54          },
55          "put": {
56            "path": "/api/device/savingMode",
57            "responses": [
58              {
59                "code": "201",
60                "description": "set the savingMode"
61              },
62              {
63                "code": "503",
64                "description": "service unavailable"
65              }
66            ]
67          }
68        }
69      ]
70    }
71  }
```

# Projekat 3 Grafana





# Projekat 3 Akcija vezana za komandu

GET <http://localhost:48080/api/v1/valuedescriptor>

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```
26     "mediaType": "text/plain"
27   },
28   {
29     "id": "3705f257-553f-4bd8-bd9a-d587a369dd6b",
30     "created": 1655380355427,
31     "description": "Fridge power consumption in watts",
32     "name": "fridge",
33     "min": "0",
34     "max": "1000",
35     "defaultValue": "0",
36     "type": "Int64",
37     "uomLabel": "fridge_watts",
38     "formatting": "%s",
39     "labels": [
40       "power_consumption",
41       "fridge"
42     ]
43   },
44   {
45     "id": "4679df97-999b-4cb4-8996-56f0c008c6dd",
46     "created": 1655380303907,
47     "description": "Furnace power consumption in watts",
48     "name": "furnace",
49     "min": "0",
50     "max": "3000",
51     "defaultValue": "0",
52     "type": "Int64",
53     "uomLabel": "furnace_watts",
54     "formatting": "%s",
55     "labels": [
56       "power_consumption",
57       "furnace"
58     ]
59   },
60 }
```



# Linkovi

Public API koji je korišćen: [Open Library Search API | Open Library](#)

Dataset za prva dva projekta: [Book Recommendation Dataset | Kaggle](#)

Dataset za treći projekat: [Smart Home Dataset with weather Information | Kaggle](#)

Na git-u, u repozitorijumu samog porjekta možete naći upustva za pokretanje projekata kao i kolekciju upita u Postman-u za testiranje rada aplikacije.