

# Odbrana projekta

▶ Slavko Petrović 17345

# Projekat I

- ▶ Podaci: Marvelovi filmovi
- ▶ [https://www.kaggle.com/datasets/minisam/marvel-movie-dataset?select=marvel\\_clean.csv](https://www.kaggle.com/datasets/minisam/marvel-movie-dataset?select=marvel_clean.csv)
- ▶ Public API: imdb
- ▶ <https://imdb-api.com/>
- ▶ Svaki mikroservis radi u okviru Docker containera

# Projekat I - mikroservisi

- ▶ 1. Movie service:
  - ▶ Tehnologija NodeJS
  - ▶ Vrsi komunikaciju sa MongoDB bazom
- ▶ 2. Gateway service:
  - ▶ Tehnologija .NET 6
  - ▶ Sinhrono pristupa prvom mikroservisu I PublicApi-u

# Projekat I - pomocna aplikacija

## ► Helper app

- Tehnologija .NET6
- Vrsi popunjavanje baze
- Pristupa gateway mikroservisu

```
var currentDir = Environment.CurrentDirectory;
using (var streamReader = new StreamReader(Path.Combine(currentDir, @"marvel_movies.csv")))
{
    using (var csvReader = new CsvReader(streamReader, CultureInfo.InvariantCulture))
    {
        var records = csvReader.GetRecords<MarvelMovies>().ToList();
        var timer = new PeriodicTimer(TimeSpan.FromSeconds(5));
        var count = 0;
        while (await timer.WaitForNextTickAsync())
        {
            var serializedObject = JsonConvert.SerializeObject(records[count]);
            var movie = new StringContent(serializedObject, Encoding.UTF8, "application/json");
            using (var httpClient = new HttpClient())
            {
                await httpClient.PostAsync($"http://host.docker.internal:5001/Gateway/postMovie", movie);
            }
            Console.WriteLine(count + " / 63");
            count++;

            if (count > records.Count - 1)
            {
                timer.Dispose();
            }
        }
    }
}
```

# Swagger UI

**gateway** 1.0 OAS3  
<http://localhost:5001/swagger/v1/swagger.json>

## Gateway

DELETE

/Gateway/deleteMovie/{title}

DELETE

/Gateway/deleteMovies

POST

/Gateway/postMovie

PUT

/Gateway/updateMovie/{title}

GET

/Gateway/getMovie/{title}

GET

/Gateway/getRandomMovie

## Schemas

MarvelMovies >

UpdateMovies >

GET /Gateway/getRandomMovie

Parameters

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5001/Gateway/getRandomMovie' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5001/Gateway/getRandomMovie
```

Server response

Code	Details
200	<div>Response body</div> <div><pre>{   "movie": {     "title": "Spider-Man: Homecoming",     "distributor": "Sony Pictures",     "releaseDateUS": "2017-07-07T00:00:00Z",     "budget": 175000000,     "openingWeekendNorthAmerica": 117027503,     "northAmerica": 334201140,     "otherTerritories": 545965784,     "worldwide": 880166924   },   "reviews": {     "imdb": "7.4/10",     "fmAffinity": "6.3/10",     "theMovieDb": "7.4/10",     "metacritic": "73/100",     "rottenTomatoes": "92/100"   } }</pre></div> <div>Response headers</div> <div><pre>content-type: application/json; charset=utf-8 date: Mon, 22 Aug 2022 22:36:04 GMT server: Kestrel transfer-encoding: chunked</pre></div>

Responses

Code	Description	Links
200	Success	No links

# Projekat II

- ▶ Prosirenje Projekta I, koriste se isti podaci
- ▶ Uvodi se izmena u gateway servisu iz prvog projekta tako sto se prosiruje POST operacija publishovanjem na Mosquitto mqtt.
- ▶ Ekuiper analizira te podatke I ukoliko je je worldwide prihod od filma veci od zadate vrednosti prosledjuje na izlazni mqtt topic.

# Projekat II - mikroservisi

- ▶ 1. Analytics microservice:
  - ▶ Tehnologija NodeJS
  - ▶ Dobija podatke poslane od strane ekuipera I upisuje ih u influxDB.
  - ▶ Pomocu gRPC protokola salje notifikaciju drugom servisu
- ▶ 2. Notification microservice
  - ▶ Tehnologija python
  - ▶ Ispisuje podatke koji mu je poslao analytics microservice

# eKuiper

Service name	Endpoint
movie	http://host.docker.internal:9081

Stream Name: movies

Stream Fields:

Name	Type
Title	string
Distributor	string
Budget	bigint
OpeningWeekendNorthAmerica	bigint
NorthAmerica	bigint
OtherTerritories	bigint
Worldwide	bigint

datasource: inputMQTT

format: json

confKey: default

type: mqtt

strictValidation: true

SQL

```
1 select * from movies where Worldwide > 629054379
```

\* Sink

[Documentation](#)

mqtt

Connection selector ?

MQTT broker address ?

tcp://broker.emqx.io:1883

MQTT topic ?

outputMQTT

MQTT ClientID ?

MQTT protocol version ?

3.1

QoS ?

Username ?

admin

Password ?

••••••

Certification path ?

Private key path ?

Root Ca path ?

Skip Certification verification ?

☐ True ☐ False

Omit if content is empty ?

☐ True ☐ False

Send single ?

☐ True ☐ False

Stream Format

json



# Analytics microservice container




The screenshot shows a Docker container interface for 'application-analytics-1' in the 'analytics' namespace, which is in a 'RUNNING' state. The terminal displays the following output:

```
> analytics@1.0.0 start
> nodemon app.js

[nodemon] 2.0.19
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
Mqtt client subscribed to the topic
mqttOutput recieved filtered movie from eKuiper : [{"Budget":139000000,"Distributor":"Sony Pictures","NorthAmerica":403706375,"OpeningWeekendNorthAmerica":114844116,"OtherTerritories":418002176,"Title":"Spider-Man","Worldwide":821708551}]
mqttOutput recieved filtered movie from eKuiper : [{"Budget":200000000,"Distributor":"Sony Pictures","NorthAmerica":373585825,"OpeningWeekendNorthAmerica":88156227,"OtherTerritories":415390628,"Title":"Spider-Man 2","Worldwide":788976453}]
```

# Notification microservice container

```
<  notifications_application notifications
RUNNING

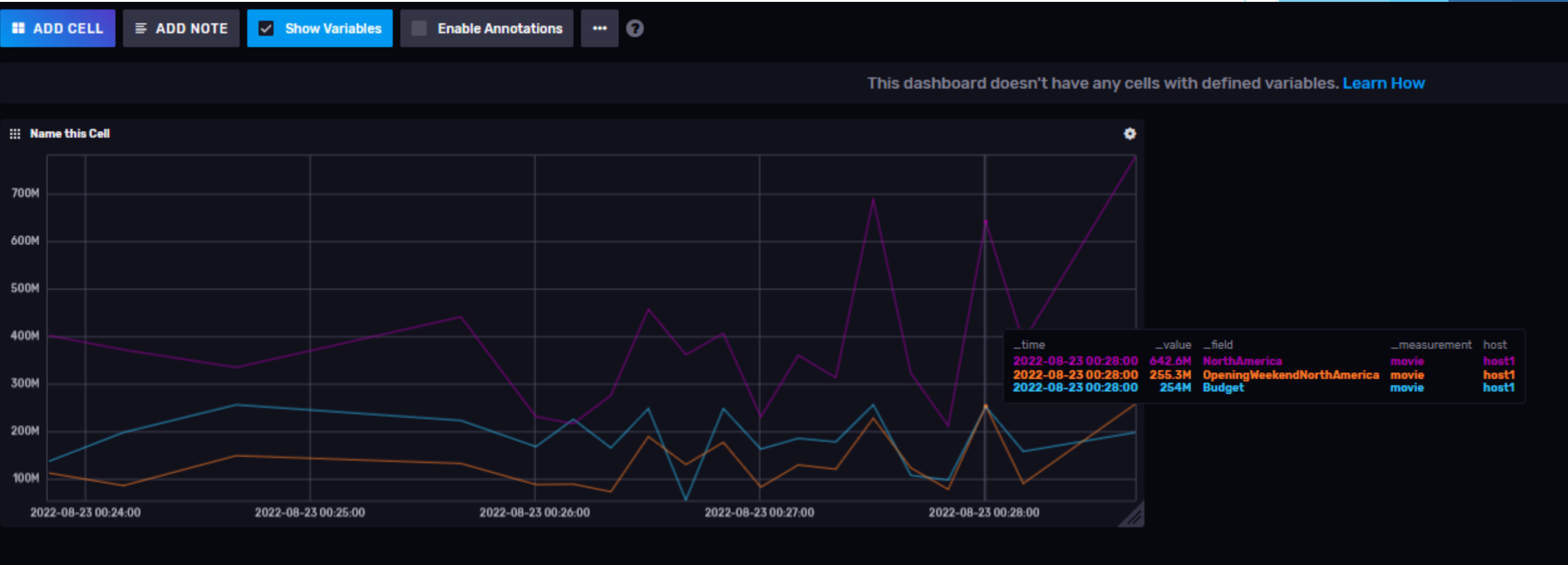
Grpc server started on port 5011
Notification! Must watch movie Spider-Man!
Title: "Spider-Man"
Distributor: "Sony Pictures"
Budget: 139000000
OpeningWeekendNorthAmerica: 114844116
NorthAmerica: 403706375
OtherTerritories: 418002176
Worldwide: 821708551

Notification! Must watch movie Spider-Man 2!
Title: "Spider-Man 2"
Distributor: "Sony Pictures"
Budget: 200000000
OpeningWeekendNorthAmerica: 88156227
NorthAmerica: 373585825
OtherTerritories: 415390628
Worldwide: 788976453

Notification! Must watch movie Spider-Man 3!
Title: "Spider-Man 3"
Distributor: "Sony Pictures"
Budget: 258000000
OpeningWeekendNorthAmerica: 151116516
NorthAmerica: 336530303
OtherTerritories: 554341323
Worldwide: 890871626

Notification! Must watch movie The Avengers!
Title: "The Avengers"
Distributor: "Walt Disney Studios Motion Pictures"
Budget: 220000000
OpeningWeekendNorthAmerica: 207438708
NorthAmerica: 623357910
OtherTerritories: 895455078
Worldwide: 1518812988
```

# InfluxDB



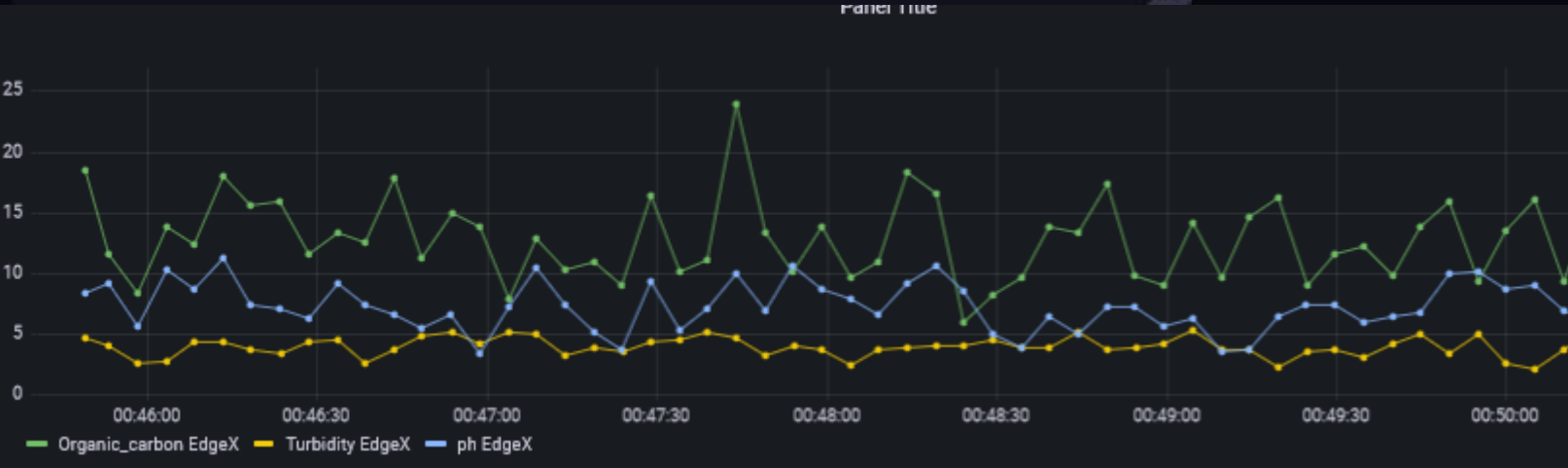
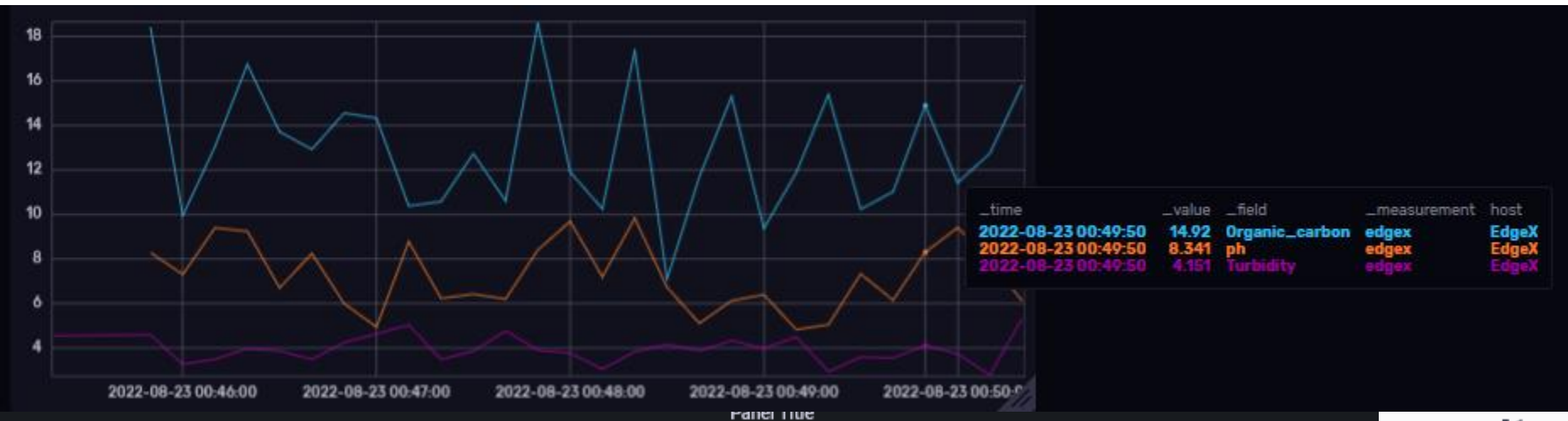
# Projekat III -podaci I pomocna aplikacija

- ▶ Podaci: mere kvaliteta vode
- ▶ <https://www.kaggle.com/datasets/adityakadiwal/water-potability>
- ▶ Svaki mikroservis radi u okviru Docker containera
- ▶ Pomocna aplikacija se nalazi u folderu sensorDataGeneration, koja cita podatke o kvalitetu vode i salje ih na EdgeX Foundry platformu
- ▶ Skripte koje se nalaze u folderu deviceCreation se koriste samo prilikom prvog startovanja na nekoj aplikaciji

# Projekat III

- ▶ 1. Visualization
  - ▶ Tehnologija NodeJS
  - ▶ Podatke koje dobija od EdgeX mqtt protokola smesta u influxDB
  - ▶ Pomocu grafane omogucena je vizuelizacija podataka
- ▶ 2. Monitoring
  - ▶ Tehnologija .NET 6
  - ▶ Podatke koje dobija od EdgeX mqtt protokola analizira I salje edgeX komandu da se promeni boja kvadratica ColorApp

# InfluxDB , Grafana I Swagger UI

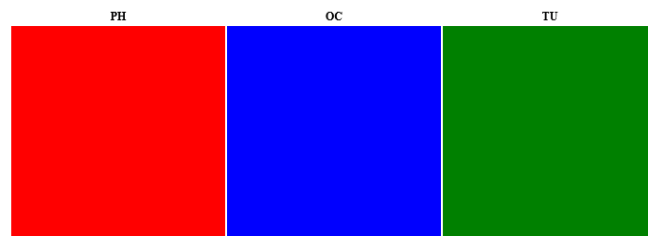


**monitoring** <sup>1.0</sup> <sup>OAS3</sup>  
<http://localhost:7049/swagger/v1/swagger.json>

### Limits

- POST** /Limits/ph/{min}/{max}
- POST** /Limits/Organic\_carbon/{min}/{max}
- POST** /Limits/Turbidity/{min}/{max}
- GET** /Limits/restartLimits

# ColorChange app



- ▶ Crvena boja znaci da je ispod dozviljenog opsega
- ▶ Plava boja znaci da je iznad dozvoljenog opsega
- ▶ Zelena boja znaci da je u granicama dozvoljenog opsega

Hvala na paznji