# INERNET STVARI I SERVISA

# SADRŽAJ


01 DATASET

02 PROJEKAT 1

03 PROJEKAT 2

04 PROJEKAT 3

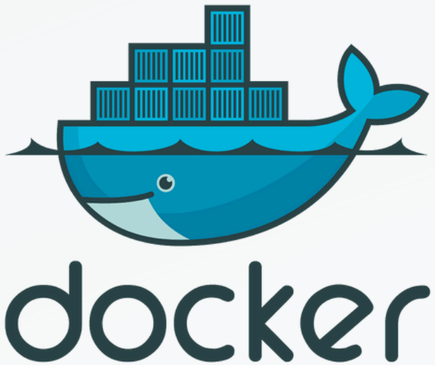
# DATA SET

- Dataset o kvalitetu vazduha
- Dataset sadrži 9357 instanci merenja
- Svaka instanca predstavlja prosečnu vrednost merenja sa pet različitih senzora
- Podaci su od marta 2004 godine do februara 2005
- Uređaji su locirani u nekoj zagađenoj oblasti u Italiji.
- https://www.kaggle.com/datasets/tawfikelmetwally/air-quality-dataset

# PROJEKAT 1- GRPC

# TEHNOLOGIJE



# DOTNET SERVIS-GRPC SERVER

## Paketi

**CsvHelper** by Josh Close — 32.0.3 / 33.0.1
A library for reading and writing CSV files. Extremely fast, flexible, and easy to use. Supports reading and writing of custom class objects.

**Google.Protobuf** by Google Inc. — 3.26.1 / 3.27.2
C# runtime library for Protocol Buffers - Google's data interchange format.

**Grpc.AspNetCore** by The gRPC Authors — 2.57.0 / 2.63.0
gRPC meta-package for ASP.NET Core

**Grpc.Net.Client** by The gRPC Authors — 2.62.0 / 2.63.0
.NET client for gRPC

**Grpc.Tools** by The gRPC Authors — 2.62.0 / 2.64.0
gRPC and Protocol Buffer compiler for C# projects

**Microsoft.EntityFrameworkCore** by Microsoft — 8.0.4 / 8.0.6
Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure Cosmos DB, MySQL, PostgreSQL, and ot...

**Microsoft.EntityFrameworkCore.Tools** by Microsoft — 8.0.4 / 8.0.6
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

**Npgsql.EntityFrameworkCore.PostgreSQL** by Shay Rojansky,Austin Drenski,Yoh Deadfall — 8.0.2 / 8.0.4
PostgreSQL/Npgsql provider for Entity Framework Core.

## Proto fajl

```
syntax = "proto3";
import "google/protobuf/duration.proto";
import "google/protobuf/timestamp.proto";
import "google/protobuf/empty.proto";

option csharp_namespace = "GrpcServer";
service AirQuality {

    rpc getDataById(DataId) returns (AirDataQuality){}
    rpc createData(AirDataQuality) returns (AirDataQuality){}
    rpc deleteData(DataId) returns (google.protobuf.Empty){}
    rpc updateData(AirDataQuality) returns (AirDataQuality){}

    rpc MinDataValueInRange(DateRange) returns (AirDataQuality){}
    rpc MaxDataValueInRange(DateRange) returns (AirDataQuality){}
    rpc AverageDataValueInRange(DateRange) returns (AverageData){}
    rpc SumDataValueInRange(DateRange) returns(SumData){}
}

message DataId{
    int32 id = 1;
}
message AirDataQuality {
      google.protobuf.Timestamp date = 1;
   google.protobuf.Duration time = 2;
   float co_gt = 3;
   int32 pt08_s1_co = 4;
   int32 nmhc_gt = 5;
   float c6h6_gt = 6;
   int32 pt08_s2_nmhc = 7;
   int32 nox_gt = 8;
   int32 pt08_s3_nox = 9;
   int32 no2_gt = 10;
   int32 pt08_s4_no2 = 11;
   int32 pt08_s5_o3 = 12;
   float t = 13;
   float rh = 14;
   float ah = 15;
   int32 id = 16;
}
message DateRange {
    string property_name = 1;
    google.protobuf.Timestamp startDate = 2;
    google.protobuf.Timestamp endDate = 3;
}
message AverageData {
    string property_name = 1;
    float averageValue = 2;
}
message SumData{
    string property_name = 1;
    float sumValue = 2;
}
```

# DBContext

```cs
using GrpcServer.Models;
using Microsoft.EntityFrameworkCore;
using System.Globalization;
namespace GrpcServer.ContextDB
{
    4 references
    public class DBContext : DbContext
    {
        protected readonly IConfiguration _configuration;
        0 references
        public DBContext  (IConfiguration configuration) { this._configuration = configuration; }
        0 references
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseNpgsql(_configuration.GetConnectionString("Database"));
        }
        0 references
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<AirQualityDataDbModel>()
             .Property(e => e.Date)
             .HasConversion(
                 v => v.ToString("yyyy-MM-dd"),
                 v => DateTime.ParseExact(v, "yyyy-MM-dd", CultureInfo.InvariantCulture));

        }
        10 references
        public DbSet<AirQualityDataDbModel> airQualities { get; set; }
    }
}
```

# AirQualityDataDbModel

```cs
    [Table("air_quality_iot1")]

    6 references
    public class AirQualityDataDbModel
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        6 references
        public int Id { get; set; }
        15 references
        public DateTime Date { get; set; }
        6 references
        public TimeSpan Time { get; set; }
        6 references
        public float CO_GT { get; set; }
        6 references
        public int PT08_S1_CO { get; set; }
        6 references
        public int NMHC_GT { get; set; }
        6 references
        public float C6H6_GT { get; set; }
        6 references
        public int PT08_S2_NMHC { get; set; }
        6 references
        public int NOx_GT { get; set; }
        6 references
        public int PT08_S3_NOx { get; set; }
        6 references
        public int NO2_GT { get; set; }
        6 references
        public int PT08_S4_NO2 { get; set; }
        6 references
        public int PT08_S5_O3 { get; set; }
        6 references
        public float T { get; set; }
        6 references
        public float RH { get; set; }
        6 references
        public float AH { get; set; }

    }
}
```

# SERVIS

```
private void loadFromCsv()
{
    var configuration = new CsvConfiguration(CultureInfo.InvariantCulture)
    { HasHeaderRecord = false };
    string csvFilePath = "/grpcServer/config/AirQuality.csv";
    using (var reader = new StreamReader(csvFilePath))
    using (var csv = new CsvReader(reader, CultureInfo.InvariantCulture))
    {
        while (csv.Read())
        {
            var record = csv.GetRecord<AirQualityData>();
            airQualityarrayFromCsv.Add(record);


        }


    }
```

```
public AirQualityService(DBContext context) {
    dbContext = context;
    if(dbContext.Database.EnsureCreated())
    {
        loadFromCsv();


            foreach (var airQuality in airQualityarrayFromCsv)
            {
                var date = DateTime.ParseExact(airQuality.Date, "dd/MM/yyyy", CultureInfo.InvariantCulture);
                var time = TimeSpan.ParseExact(airQuality.Time, "hh\\:mm\\:ss", CultureInfo.InvariantCulture);


                var airQualityDataDbModel = new AirQualityDataDbModel
                {
                    Date = date,
                    Time = time,
                    CO_GT = airQuality.CO_GT,
                    PT08_S1_CO = airQuality.PT08_S1_CO,
                    NMHC_GT = airQuality.NMHC_GT,
                    C6H6_GT = airQuality.C6H6_GT,
                    PT08_S2_NMHC = airQuality.PT08_S2_NMHC,
                    NOx_GT = airQuality.NOx_GT,
                    PT08_S3_NOx = airQuality.PT08_S3_NOx,
                    NO2_GT = airQuality.NO2_GT,
                    PT08_S4_NO2 = airQuality.PT08_S4_NO2,
                    PT08_S5_O3 = airQuality.PT08_S5_O3,
                    T = airQuality.T,
                    RH = airQuality.RH,
                    AH = airQuality.AH
                };

                context.airQualities.Add(airQualityDataDbModel);


            context.SaveChanges();
```

```
3 references
public override async Task<AirDataQuality> getDataById(DataId request, ServerCallContext context)
{
    try
    {
        int id = request.Id;
        var airDataQuality = dbContext.airQualities.FirstOrDefault(aDQ => aDQ.Id == id);
        if (airDataQuality != null)
        {
            DateTime utcDateTime = airDataQuality.Date.ToUniversalTime();

            return await Task.FromResult(new AirDataQuality
            {
                Id = airDataQuality.Id,
                Date = Timestamp.FromDateTime(utcDateTime),
                Time = Duration.FromTimeSpan(airDataQuality.Time),
                CoGt = airDataQuality.CO_GT,
                Pt08S1Co = airDataQuality.PT08_S1_CO,
                NmhcGt = airDataQuality.NMHC_GT,
                C6H6Gt = airDataQuality.C6H6_GT,
                Pt08S2Nmhc = airDataQuality.PT08_S2_NMHC,
                NoxGt = airDataQuality.NOx_GT,
                Pt08S3Nox = airDataQuality.PT08_S3_NOx,
                No2Gt = airDataQuality.NO2_GT,
                Pt08S4No2 = airDataQuality.PT08_S4_NO2,
                Pt08S5O3 = airDataQuality.PT08_S5_O3,
                T = airDataQuality.T,
                Rh = airDataQuality.RH,
                Ah = airDataQuality.AH
            });
        }
        else
        {
            return null;
        }
    }
    catch (Exception ex)
    {

        throw new RpcException(new Status(StatusCode.Internal, ex.Message));
    }
}
```

Ideja je prvi put da pri inicijalizaciji servisa da se podaci ucitaju iz csv fajla i upišu u bazu.

# SPRINGBOOT SERVIS- GRPC KLIJENT

## pom.xml - dependeny

```xml
</dependency>
<dependency>
    <groupId>io.grpc</groupId>
    <artifactId>grpc-netty</artifactId>
    <version>1.62.2</version>
</dependency>
<dependency>
    <groupId>io.grpc</groupId>
    <artifactId>grpc-protobuf</artifactId>
    <version>1.62.2</version>

</dependency>
<dependency>
    <groupId>com.google.protobuf</groupId>
    <artifactId>protobuf-java-util</artifactId>
    <version>3.25.3</version>
</dependency>
<dependency>
    <groupId>io.grpc</groupId>
    <artifactId>grpc-stub</artifactId>
    <version>1.62.2</version>
</dependency>
```

## RestClient

```java
@Component  2 usages  VVeljovic
public class RestClient {
private ManagedChannel channel;  2 usages
private AirQualityGrpc.AirQualityBlockingStub blockingStub;  9 usages
public RestClient()  VVeljovic
{
    this.channel = ManagedChannelBuilder.forAddress( name: "dotnetapp",  port: 8080).usePlaintext().build();
    this.blockingStub = AirQualityGrpc.newBlockingStub(channel);
}
    public AirDataQuality getDataById(int id)  1 usage  VVeljovic
    {
        DataId dataId = DataId.newBuilder().setId(id).build();
        return blockingStub.getDataById(dataId);
    }
    public AirDataQuality createData(AirDataQuality data) { return blockingStub.createData(data); }
```

# RestController

```java
@RequestMapping(path="airquality")
public class AirQualityController {
    private final RestClient restClient;  9 usages

    @Autowired  VVeljovic *
    public AirQualityController(RestClient restClient) {
        this.restClient = restClient;
    }

    @GetMapping("/getData/{id}")  VVeljovic
    public ResponseEntity<String> getDataById(@PathVariable Integer id) {

        AirDataQuality data = restClient.getDataById(id);


        try{
            String jsonData = JsonFormat.printer().print(data);
            long seconds = data.getTime().getSeconds();
            String formattedTime = convertSecondsToHHMMSS(seconds);
            jsonData = jsonData.replace( target: "\"" + seconds + "s\"",  replacement: "\"" + formattedTime + "\"");
            return ResponseEntity.ok(jsonData);
        }
        catch (Exception e)
        {
            e.printStackTrace();
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Greška prilikom konverzije objekta u json.");
        }
    }
```

# DOCKER FAJLOVI

## Spring

```
FROM eclipse-temurin:17-jdk-focal

WORKDIR /app

COPY .mvn/ .mvn
COPY mvnw pom.xml ./
RUN ./mvnw dependency:go-offline

COPY src ./src

CMD ["./mvnw", "spring-boot:run"]
```

## .NET

Dockerfile

```
FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base
WORKDIR /app
EXPOSE 8080

FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
WORKDIR /src
COPY ["GrpcServer/GrpcServer.csproj", "GrpcServer/"]
RUN dotnet restore "GrpcServer/GrpcServer.csproj"
COPY . .
WORKDIR "/src/GrpcServer"
RUN dotnet build "GrpcServer.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "GrpcServer.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "GrpcServer.dll"]

```

# DOCKER COMPOSE

docker-compose.yaml

```yaml
version: '3.8'

services:
  postgress:
    container_name: postgress
    image: postgres
    restart: always
    environment:
      POSTGRES_DB: IOT1
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: Veljko22!!!
    ports:
      - "5433:5432"
    networks:
      - iot
  dotnetapp:
    container_name: dotnet
    build:
      context: C:\Users\veljk\OneDrive\Desktop\Cetvrta godina\IOT\IOT-AirQuality\GrpcServer
      dockerfile: Dockerfile
    ports:
      - "8080:8080"
    depends_on:
      - postgress
    networks:
      - iot
    volumes:
      - ./AirQuality.csv:/grpcServer/config/AirQuality.csv
  springbootapp:
      container_name: springboot
      build:
        context: C:\Users\veljk\OneDrive\Desktop\Cetvrta godina\IOT\IOT-AirQuality\RestServer
        dockerfile: Dockerfile
      ports:
        - "5050:8080"
      networks:
        - iot
networks:
  iot:
    name: iot
```

```
C:\Users\veljk\OneDrive\Desktop\Cetvrta godina\IOT\IOT-AirQuality>docker compose up -d
time="2024-06-27T21:30:55+02:00" level=warning msg="C:\\Users\\veljk\\OneDrive\\Desktop\\Cetvrta godina\\IOT\\IOT-AirQua
lity\\docker-compose.yaml: `version` is obsolete"
[+] Running 3/3
 ✔ Container postgress   Started                                                                                    2.2s
 ✔ Container springboot  Started                                                                                    2.2s
 ✔ Container dotnet      Started                                                                                    4.0s

C:\Users\veljk\OneDrive\Desktop\Cetvrta godina\IOT\IOT-AirQuality>docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS
       NAMES
5fab62ad956c   iot-airquality-springbootapp   "/__cacert_entrypoin…"   57 seconds ago   Up 5 seconds    0.0.0.0:5050->808
0/tcp   springboot
a5f778126fff   iot-airquality-dotnetapp       "dotnet GrpcServer.d…"   2 days ago       Up 3 seconds    0.0.0.0:8080->808
0/tcp   dotnet
8350edbbf50e   postgres                       "docker-entrypoint.s…"   12 days ago      Up 5 seconds    0.0.0.0:5433->543
2/tcp   postgress

C:\Users\veljk\OneDrive\Desktop\Cetvrta godina\IOT\IOT-AirQuality>
```

# TESTIRANJE PREKO POSTMANA

AirQuality / **createData**

POST http://localhost:5050/airquality/createData

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
  "date": "2055-04-25",
  "time": "12:00:00",
  "co_gt": 1.1,
  "pt08_s1_co": 200,
  "nmhc_gt": 50,
  "c6h6_gt": 5.0,
  "pt08_s2_nmhc": 150,
  "nox_gt": 100,
  "pt08_s3_nox": 300,
  "no2_gt": 80,
  "pt08_s4_no2": 250,
  "pt08_s5_o3": 350,
  "t": 25.0,
  "rh": 50.0,
  "ah": 0.5
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
{
  "date": "2055-04-25T00:00:00Z",
  "time": "12:00:00",
  "coGt": 1.1,
  "pt08S1Co": 200,
  "nmhcGt": 50,
  "c6h6Gt": 5.0,
  "pt08S2Nmhc": 150,
  "noxGt": 100,
  "pt08S3Nox": 300,
  "no2Gt": 80,
  "pt08S4No2": 250,
  "pt08S5O3": 350,
  "t": 25.0,
  "rh": 50.0,
  "ah": 0.5,
  "id": 9370
}
```

AirQuality / **getData**

GET http://localhost:5050/airquality/getData/9370

Params Authorization Headers (7) Body Pre-request Script Tests Setting

Headers 7 hidden

| Key | Val |
|---|---|
| Key | Val |

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
{
  "date": "2055-04-25T00:00:00Z",
  "time": "12:00:00",
  "coGt": 1.1,
  "pt08S1Co": 200,
  "nmhcGt": 50,
  "c6h6Gt": 5.0,
  "pt08S2Nmhc": 150,
  "noxGt": 100,
  "pt08S3Nox": 300,
  "no2Gt": 80,
  "pt08S4No2": 250,
  "pt08S5O3": 350,
  "t": 25.0,
  "rh": 50.0,
  "ah": 0.5,
  "id": 9370
}
```

AirQuality / **updateData**

PUT http://localhost:5050/airquality/updateData

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
  "date": "2024-04-30",
  "time": "13:01:00",
  "co_gt": 0.5,
  "pt08_s1_co": 11,
  "nmhc_gt": 50,
  "c6h6_gt": 5.0,
  "pt08_s2_nmhc": 150,
  "nox_gt": 100,
  "pt08_s3_nox": 300,
  "no2_gt": 80,
  "pt08_s4_no2": 250,
  "pt08_s5_o3": 350,
  "t": 25.0,
  "rh": 50.0,
  "ah": 0.5,
  "id": 9370
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

```
{
  "date": "2024-04-30T00:00:00Z",
  "time": "13:01:00",
  "coGt": 0.5,
  "pt08S1Co": 11,
  "nmhcGt": 50,
  "c6h6Gt": 5.0,
  "pt08S2Nmhc": 150,
  "noxGt": 100,
  "pt08S3Nox": 300,
  "no2Gt": 80,
  "pt08S4No2": 250,
  "pt08S5O3": 350,
  "t": 25.0,
  "rh": 50.0,
  "ah": 0.5,
  "id": 9370
}
```

# TESTIRANJE PREKO POSTMANA

AirQuality / **deleteData**

DELETE | http://localhost:5050/airquality/deleteData/9370

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL

This request does not h

Body | Cookies | Headers (5) | Test Results

Pretty | Raw | Preview | Visualize | Text

```
1  Data with ID 9370 deleted successfully.
```

AirQuality / **min**

GET | http://localhost:5050/airquality/minDataInRange?start=2005-04-02&end=2005-04-03&property=CO_GT

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings

Query Params

| Key | Value |
|---|---|
| start | 2005-04-02 |
| end | 2005-04-03 |
| property | CO_GT |
| Key | Value |

Body | Cookies | Headers (5) | Test Results

Pretty | Raw | Preview | Visualize | Text


```
{
  "date": "2005-04-03T00:00:00Z",
  "time": "05:00:00",
  "coGt": 0.3,
  "pt08S1Co": 811,
  "nmhcGt": -200,
  "c6h6Gt": 1.0,
  "pt08S2Nmhc": 494,
  "noxGt": 60,
  "pt08S3Nox": 1118,
  "no2Gt": 46,
  "pt08S4No2": 872,
  "pt08S5O3": 470,
  "t": 9.6,
  "rh": 51.0,
  "ah": 0.6082,
  "id": 9325
}
```

AirQuality / **max**

GET http://localhost:5050/airquality/maxDataInRange?start=2005-04-02&end=2005-04-03&property=CO_GT

Params • Authorization Headers (7) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text


```
{
  "date": "2005-04-03T00:00:00Z",
  "time": "19:00:00",
  "coGt": 2.7,
  "pt08S1Co": 1248,
  "nmhcGt": -200,
  "c6h6Gt": 11.1,
  "pt08S2Nmhc": 1018,
  "noxGt": 367,
  "pt08S3Nox": 599,
  "no2Gt": 181,
  "pt08S4No2": 1289,
  "pt08S5O3": 1167,
  "t": 19.9,
  "rh": 33.0,
  "ah": 0.7608,
  "id": 9339
}
```


GET http://localhost:5050/airquality/getAvgValue?start=2005-04-02&end=2005-04-03&property=CO_GT

Params • Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

| Key | Value | De |
|---|---|---|
| start | 2005-04-02 | |
| end | 2005-04-03 | |
| property | CO_GT | |
| Key | Value | |

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize


```
{
  "propertyName": "CO_GT",
  "averageValue": 0.9979167
}
```


AirQuality / **sum**

GET http://localhost:5050/airquality/getSumValue?start=2005-04-02&end=2005-04-03&property=NMHC_GT

Params • Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

| Key | Value |
|---|---|
| start | 2005-04-02 |
| end | 2005-04-03 |
| property | NMHC_GT |
| Key | Value |

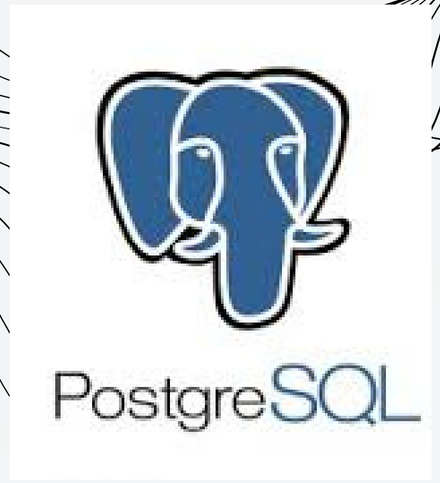
Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text


```
{
  "propertyName": "NMHC_GT",
  "sumValue": -9600.0
}
```

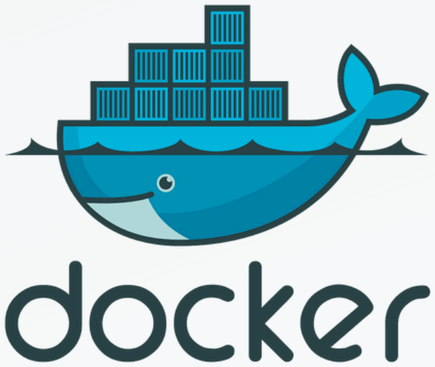
# PROJEKAT 2- MQTT

# TEHNOLOGIJE










# DOCKER COMPOSE

```yaml
services:
  postgres:
    container_name: postgres
    image: postgres
    restart: always
    environment:
      POSTGRES_DB: IOT
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: Veljko22!!!
    ports:
      - "5433:5432"
    networks:
      - iotdva

  mosquitto:
    container_name: mosquitto_container
    image: eclipse-mosquitto
    ports:
      - "1883:1883"
      - "9001:9001"
    networks:
      - iotdva
    volumes:
      - ./mosquitto.conf:/mosquitto/config/mosquitto.conf

  manager:
    image: emqx/ekuiper-manager:1.8.0
    container_name: manager
    ports:
      - "9082:9082"
    environment:
      DEFAULT_EKUIPER_ENDPOINT: "http://ekuiper:9081"
    networks:
      - iotdva
```

```yaml
  ekuiper:
    image: lfedge/ekuiper:1.8.0
    ports:
      - "9081:9081"
    container_name: ekuiper
    hostname: ekuiper
    volumes:
      - ./eKuiper/data/init.json:/kuiper/data/init.json:ro
    environment:
      MQTT_SOURCE__DEFAULT__SERVER: "tcp://mosquitto_container:1883"
    networks:
      - iotdva

  analytics:
    container_name: analytics_container
    build:
      context: Analytics
      dockerfile: Dockerfile
    ports:
      - "7070:7070"
    networks:
      - iotdva
    depends_on:
      - manager
      - mosquitto
      - ekuiper
```

```yaml
  sensor:
    container_name: sensor_container
    build:
      context: Sensor
      dockerfile: Dockerfile
    ports:
      - "6060:6060"
    networks:
      - iotdva
    depends_on:
      - postgres
      - mosquitto
    volumes:
      - ./AirQuality.csv:/sensor/config/AirQuality.csv

  springbootapp:
    container_name: event_info_container
    build:
      context: C:/Users/veljk/OneDrive/Desktop/Cetvrta godina/IOT2/RestApi
      dockerfile: Dockerfile
    ports:
      - "8080:8080"
    depends_on:
      - postgres
      - mosquitto
      - ekuiper
    networks:
      - iotdva

networks:
  iotdva:
    name: iotdva
```

```
PS C:\Users\veljk\OneDrive\Desktop\Cetvrta godina\IOT2> docker compose up -d
[+] Running 7/7
 ✓ Container postgres                Started
 ✓ Container manager                 Started
 ✓ Container ekuiper                 Started
 ✓ Container mosquitto_container     Started
 ✓ Container event_info_container    Started
 ✓ Container analytics_container     Started
 ✓ Container sensor_container        Started
```

# SENSOR MIKROSERVIS

```
1 reference
public void ReadFromDb()
{
    var connectionString = "Server=postgres;Port=5432;Database=IOT;Username=postgres;password=Veljko22!!!";

    var queryString = "SELECT * FROM air_quality_iot WHERE \"Id\" <= @Id;";
    using (var connection = new NpgsqlConnection(connectionString))
    {
        connection.Open();
        using (var cmd = new NpgsqlCommand(queryString, connection))
        {
            cmd.Parameters.AddWithValue("Id", 1000);
            using (var reader = cmd.ExecuteReader())
            {
                while (reader.Read())
                {

                    var airQualityData = new AirQualityDataDbModel
                    {
                        Id = reader.GetInt32(reader.GetOrdinal("id")),
                        Date = DateOnly.ParseExact(reader.GetString(reader.GetOrdinal("Date")), "yyyy-MM-dd", CultureInfo.InvariantCulture)
                        Time = reader.GetTimeSpan(reader.GetOrdinal("Time")),
                        CO_GT = reader.GetFloat(reader.GetOrdinal("co_gt")),
                        PT08_S1_CO = reader.GetFloat(reader.GetOrdinal("PT08_S1_CO")),
                        NMHC_GT = reader.GetFloat(reader.GetOrdinal("NMHC_GT")),
                        C6H6_GT = reader.GetFloat(reader.GetOrdinal("C6H6_GT")),
                        PT08_S2_NMHC = reader.GetFloat(reader.GetOrdinal("PT08_S2_NMHC")),
                        NOx_GT = reader.GetFloat(reader.GetOrdinal("NOx_GT")),
                        PT08_S3_NOx = reader.GetFloat(reader.GetOrdinal("PT08_S3_NOx")),
                        NO2_GT = reader.GetFloat(reader.GetOrdinal("NO2_GT")),
                        PT08_S4_NO2 = reader.GetFloat(reader.GetOrdinal("PT08_S4_NO2")),
                        PT08_S5_O3 = reader.GetFloat(reader.GetOrdinal("PT08_S5_O3")),
                        T = 100,
                        RH = reader.GetFloat(reader.GetOrdinal("RH")),
                        AH = reader.GetFloat(reader.GetOrdinal("AH"))
                    };

                    Thread.Sleep(1000);
                    SendToTopic("air_topic", airQualityData);
                }

            }

        }
```

```
1 reference
public  void SendToTopic(string topic, AirQualityDataDbModel airQualityData)
{

    string brokerAddress = "mosquitto";
    int brokerPort = 1883;

    string jsonData = JsonConvert.SerializeObject(airQualityData);

    string clientId = Guid.NewGuid().ToString();


    MqttClient mqttClient = new MqttClient(brokerAddress, brokerPort, false, null, null, MqttSslProtocols.None);



    try
    {

        if (!mqttClient.IsConnected)
        {
            mqttClient.Connect(clientId);
        }
        mqttClient.Publish(topic, Encoding.UTF8.GetBytes(jsonData), MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, false);
        Console.WriteLine($"Message '{jsonData}' published to topic '{topic}'");




    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error: {ex.Message}");
    }
}
```

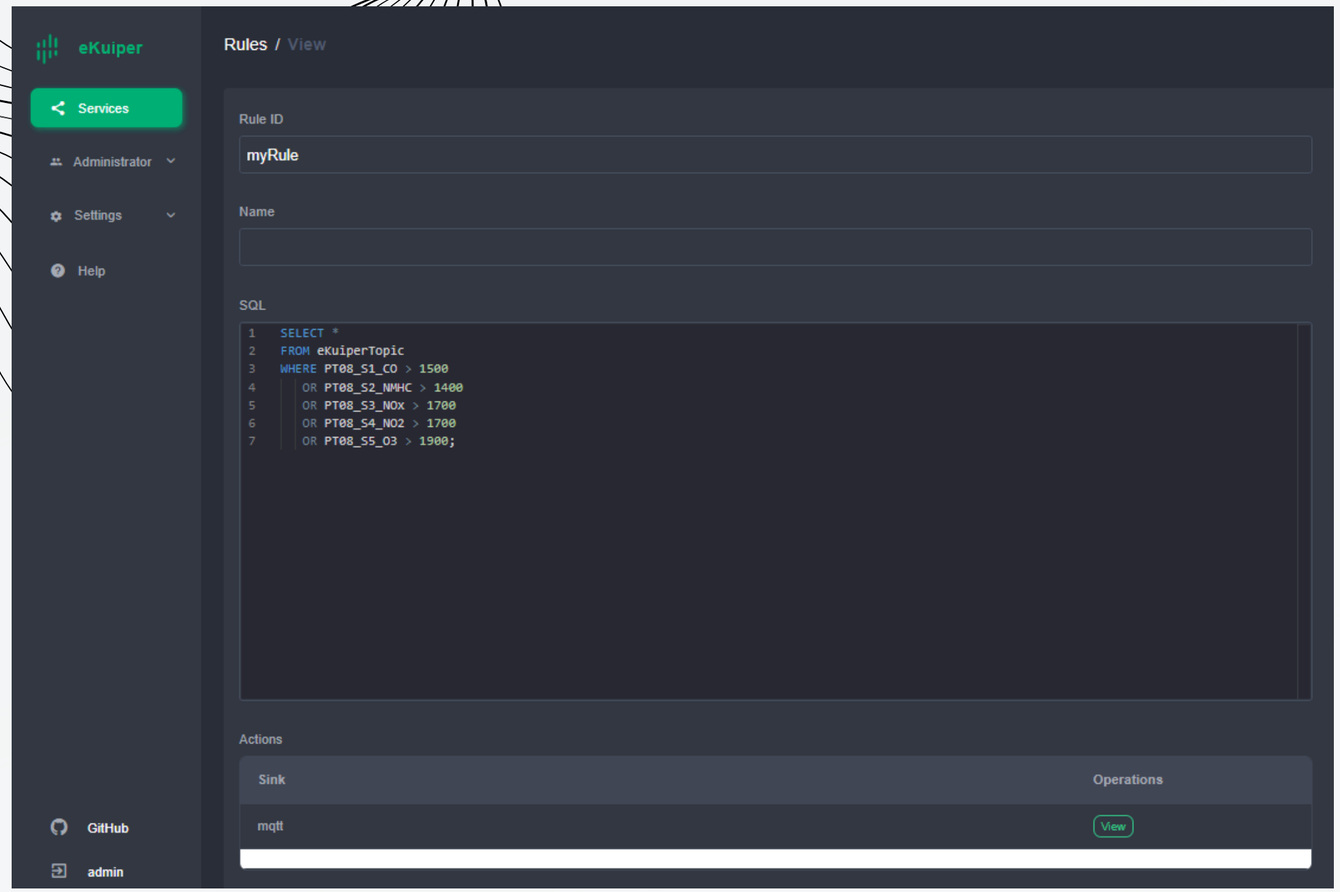
# ANALYTICS MIKROSERVIS

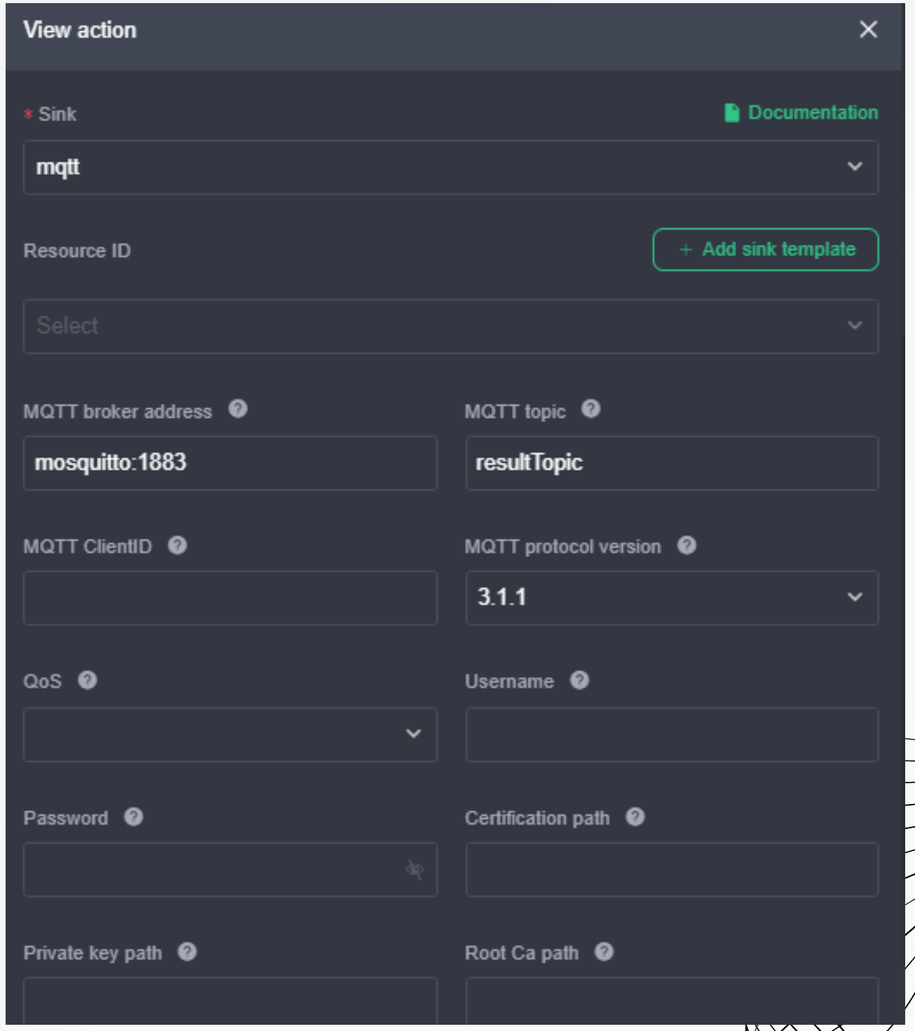
```
private MqttClient mqttClient;
private List<string> messages;
private string[] topics = { "air_topic", "resultTopic" };
private float PT08_S1_CO = 1500;
private float PT08_S2_NMHC = 1400;
private float PT08_S3_NOx = 1700;
private float PT08_S4_NO2 = 1700;
private float PT08_S5_O3 = 1900;
1 reference
public MsAnalytics()
{
    this.getDataFromTopic();
    messages = new List<string>();
}
1 reference
public void getDataFromTopic()
{
    string brokerAddress = "mosquitto";
    int brokerPort = 1883;
    string clientId = Guid.NewGuid().ToString();
    this.mqttClient = new MqttClient(brokerAddress, brokerPort,false,null,null,MqttSslProtocols.None);
    mqttClient.MqttMsgPublishReceived += SendDataToeKuiperTopic;

    try
    {
        mqttClient.Connect(clientId);
        mqttClient.Subscribe(topics, new byte[] { MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE, MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE });
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
}
```

```
5 references
private string FormMeasurementMessage(string pollutant, float currentValue, float normalValue)
{
    float change = currentValue - normalValue;
    float percentageChange = (change / normalValue) * 100;
    string direction = change > 0 ? "increase" : "decrease";
    string measurementMessage = $" Measured {pollutant} value is {currentValue} which is a {Math.Abs(percentageChange):0.00}% {direction} from the normal value of {normalValue}.";
    return measurementMessage;
}
```

```
1 reference
private  void SendDataToeKuiperTopic(object sender, MqttMsgPublishEventArgs e)
{
    string topic = e.Topic;
    string message = Encoding.UTF8.GetString(e.Message);
    if (topic == topics[0])
    {

      Console.WriteLine(message);
       this.mqttClient.Publish("eKuiperTopic", Encoding.UTF8.GetBytes(message), MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE, false);
    }
    else if (topic == topics[1])
    {
       Console.WriteLine($"From '{topic}' i got message {message}");
       JObject airDataQuality = JObject.Parse(message);
       float currentPT08_S1_CO = (float)airDataQuality["PT08_S1_CO"];
       float currentPT08_S2_NMHC = (float)airDataQuality["PT08_S2_NMHC"];
       float currentPT08_S3_NOx = (float)airDataQuality["PT08_S3_NOx"];
       float currentPT08_S4_NO2 = (float)airDataQuality["PT08_S4_NO2"];
       float currentPT08_S5_O3 = (float)airDataQuality["PT08_S5_O3"];
       string dateTime = (string)airDataQuality["Date"];
       string time = (string)airDataQuality["Time"];

       string date = dateTime.Split('T')[0];
       string formattedDate = DateTime.Parse(date).ToString("MM/dd/yyyy");
       string messageToPublic = $"On {formattedDate} at {time}, the following air quality measurements were taken:";


       messageToPublic += FormMeasurementMessage("carbon monoxide (CO)", currentPT08_S1_CO, this.PT08_S1_CO);
       messageToPublic += FormMeasurementMessage("non-methane hydrocarbons (NMHC)", currentPT08_S2_NMHC, this.PT08_S2_NMHC);
       messageToPublic += FormMeasurementMessage("nitrogen oxides (NOx)", currentPT08_S3_NOx, this.PT08_S3_NOx);
       messageToPublic += FormMeasurementMessage("nitrogen dioxide (NO2)", currentPT08_S4_NO2, this.PT08_S4_NO2);
       messageToPublic += FormMeasurementMessage("ozone (O3)", currentPT08_S5_O3, this.PT08_S5_O3);

       Console.WriteLine(messageToPublic);



       this.mqttClient.Publish("eventInfoTopic", Encoding.UTF8.GetBytes(messageToPublic), MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE, false);
    }
}
```

# EKUIPER


eKuiper
Rules / View
Services
Administrator
Settings
Help
GitHub
admin
Rule ID
myRule
Name
SQL
SELECT *
FROM eKuiperTopic
WHERE PT08_S1_CO > 1500
OR PT08_S2_NMHC > 1400
OR PT08_S3_NOx > 1700
OR PT08_S4_NO2 > 1700
OR PT08_S5_O3 > 1900;
Actions
Sink
Operations
mqtt
View


View action
* Sink
Documentation
mqtt
Resource ID
+ Add sink template
Select
MQTT broker address
mosquitto:1883
MQTT topic
resultTopic
MQTT ClientID
MQTT protocol version
3.1.1
QoS
Username
Password
Certification path
Private key path
Root Ca path

# EVENT INFO MIKROSERVIS

## MQTT Config fajl

```java
@Configuration  VVeljovic
public class MqttConfig implements MqttCallback {
    @Value("tcp://mosquitto:1883")
    private String broker;

    @Value("springBootClient")
    private String clientId;

    @Value("eventInfoTopic")
    private String topic;

    private List<String> messagesFromTopic =new ArrayList<>();  2 usages
    @Bean  VVeljovic
    public MqttClient mqttClient() throws  MqttException {
        MqttClient mqttClient = new MqttClient(broker, clientId, new MemoryPersistence());
        MqttConnectOptions mqttConnectOptions = new MqttConnectOptions();
        mqttConnectOptions.setCleanSession(true);
        mqttClient.setCallback(this);
        mqttClient.connect(mqttConnectOptions);


        mqttClient.subscribe(topic);
        return mqttClient;
    }
    @Override  VVeljovic
    public void deliveryComplete(IMqttDeliveryToken iMqttDeliveryToken) {

    }

    @Override  2 usages  VVeljovic
    public void connectionLost(Throwable throwable) {

    }

    @Override  VVeljovic
    public void messageArrived(String s, MqttMessage mqttMessage) throws Exception {
        String payload = new String(mqttMessage.getPayload());
        messagesFromTopic.add(payload);
        System.out.println("Poruka primljena sa topica: " + topic + ", sadržaj: " + payload);
    }

    public List<String> getMessageList() { return messagesFromTopic; }
```

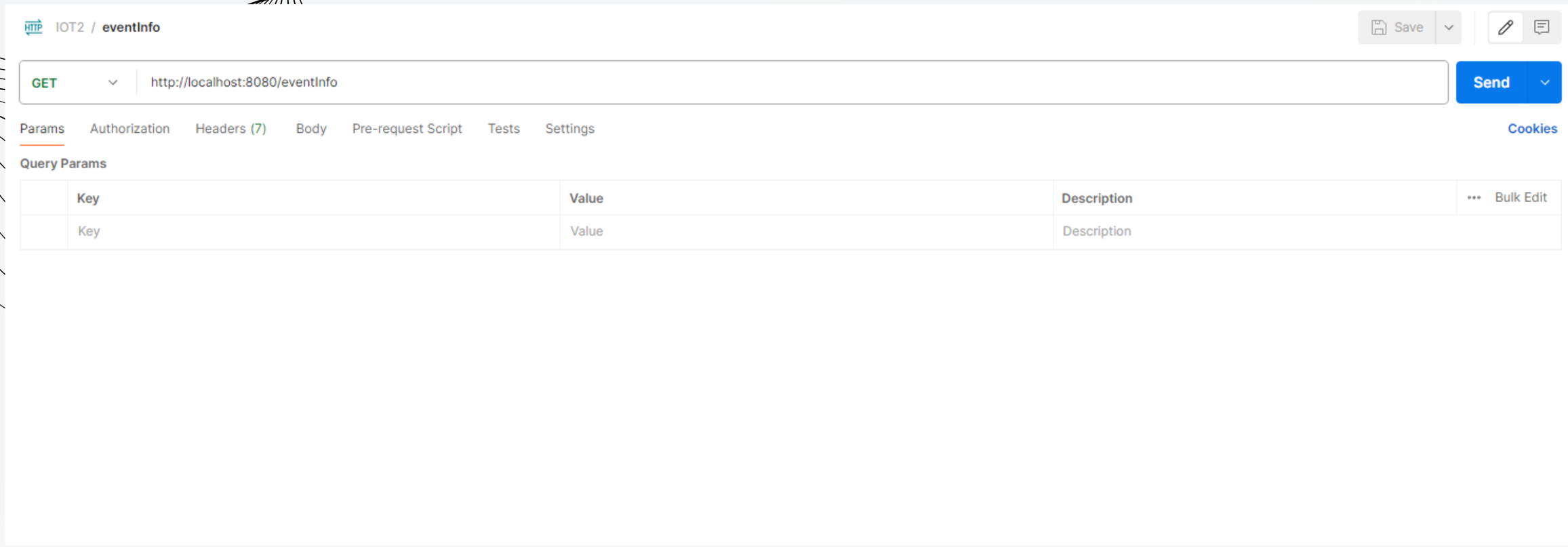
## application.properties

```
spring.application.name=RestApi
mqtt.broker=tcp://mosquitto:1883
mqtt.clientId=springBootClient
mqtt.topic=eventInfoTopic
```

## EventInfoController

```java
@RestController  VVeljovic
@RequestMapping(path = "eventInfo")
public class EventInfoController {

    private final MqttConfig config;  2 usages
    @Autowired  VVeljovic
    public EventInfoController(MqttConfig config) {this.config = config;}
    @GetMapping("")  VVeljovic
    public List<String> getData()
    {

        return config.getMessageList();
    }
}
```

# TESTIRANJE PREKO POSTMANA



# PROJEKAT 3

# TEHNOLOGIJE

influxdb

Nest JS

PostgreSQL

Microsoft .NET

spring boot

mosquitto

docker

eKuiper

# DOCKER COMPOSE

```yaml
services:
  mosquitto:
    container_name: mosquitto_container2
    image: eclipse-mosquitto
    ports:
      - "1883:1883"
      - "9001:9001"
    networks:
      - iottri
    volumes:
      - ./mosquitto.conf:/mosquitto/config/mosquitto.conf

  eKuipermanager:
    image: emqx/ekuiper-manager:1.8.0
    container_name: eKuipermanager_container
    ports:
      - "9082:9082"
    environment:
      DEFAULT_EKUIPER_ENDPOINT: "http://ekuiper:9081"
    networks:
      - iottri

  ekuiper:
    image: lfedge/ekuiper:1.8.0
    ports:
      - "9081:9081"
    container_name: ekuiper_container
    hostname: ekuiper
    volumes:
      - ./eKuiper/data/init.json:/kuiper/data/init.json:ro
    environment:
      MQTT_SOURCE__DEFAULT__SERVER: "tcp://mosquitto_container2:1883"
    networks:
      - iottri
```

```yaml
nats:
  image: nats
  ports:
    - "8222:8222"
    - "4222:4222"
  container_name: nats_container
  networks:
  - iottri

influxdb:
  image: influxdb:2
  container_name: influxdb
  ports:
    - "8086:8086"
  environment:
    - DOCKER_INFLUXDB_INIT_MODE=setup
    - DOCKER_INFLUXDB_INIT_USERNAME=veljko
    - DOCKER_INFLUXDB_INIT_PASSWORD=InternetOfThings
    - DOCKER_INFLUXDB_INIT_ORG=IotOrg
    - DOCKER_INFLUXDB_INIT_BUCKET=AirQualityData
    - DOCKER_INFLUXDB_INIT_RETENTION=0
    - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=QTcEt8MO8IIovqZinJr9POCqUc8IljNe-qmsGxjMvoJhuX1KsHtaEcbk4ejmfbHCurD_7nTjG7WpKDl3KB7v3g==
  networks:
    - iottri
grafana:
  image: grafana/grafana-oss:8.4.3
  depends_on:
    - influxdb
  ports:
    - "3000:3000"
  environment:
    GF_SECURITY_ADMIN_USER: ${GF_SECURITY_ADMIN_USER}
    GF_SECURITY_ADMIN_PASSWORD: ${GF_SECURITY_ADMIN_PASSWORD}
  networks:
    - iottri
```

```yaml
sensor:
  container_name: sensor_container_iot3
  build:
    context: Sensor
    dockerfile: Dockerfile
  ports:
    - "6060:6060"
  networks:
    - iottri
  depends_on:
    - mosquitto
  volumes:
    - ./AirQuality.csv:/sensor/config/AirQuality.csv
filter:
  container_name: filter_container
  build:
    context: Filter
    dockerfile: Dockerfile
  ports:
    - "6061:6061"
  networks:
    - iottri
  depends_on:
    - mosquitto
dashboard:
  container_name: dashboard_container
  build:
    context: dashboard
    dockerfile: Dockerfile
  ports:
    - "8080:8080"
  depends_on:
    - mosquitto
  networks:
    - iottri
```

```yaml
  command:
    container_name: command_container
    build:
      context: command
      dockerfile: Dockerfile
    ports:
      - "3001:3001"
    depends_on:
      - mosquitto
    networks:
      - iottri
networks:
  iottri:
    name: iottri
```

# COMMAND MIKROSERVIS-DOKER FAJL

```
1
2   FROM node:18
3
4
5   WORKDIR /usr/src/app
6
7
8   COPY package*.json ./
9
10
11  RUN npm install
12
13
14  COPY . .
15
16
17
18  RUN npm run build
19
20  EXPOSE 3001
21
22  CMD ["npm", "run", "start:prod"]
```

# SENSOR MIKROSERVIS

```
using uPLibrary.Networking.M2Mqtt.Messages;
List<AirQualityData> airQualityDataList = new List<AirQualityData>();
1 reference
void loadFromCsv()
{
    var configuration = new CsvConfiguration(CultureInfo.InvariantCulture)
    { HasHeaderRecord = false };
    string csvFilePath = "/sensor/config/AirQuality.csv";
    using(var reader = new StreamReader(csvFilePath))
    using(var csv = new CsvReader(reader,CultureInfo.InvariantCulture))
    {
        while(csv.Read())
        {
            var record = csv.GetRecord<AirQualityData>();
            airQualityDataList.Add(record);


        }
    }
}
loadFromCsv();
foreach(var airQualityData in airQualityDataList)
{
    string jsonData = JsonConvert.SerializeObject(airQualityData);
    sendToTopic(jsonData, "eKuiperTopic");
    Thread.Sleep(500);
}
1 reference
void sendToTopic(String jsonData, String topic)
{
    string brokerAddress = "mosquitto";
    int port = 1883;
    string clientId = Guid.NewGuid().ToString();
    MqttClient mqttClient = new MqttClient(brokerAddress, port, false, null, null, MqttSslProtocols.None);
    try
    {
        if(!mqttClient.IsConnected)
        {
            mqttClient.Connect(clientId);
        }
        mqttClient.Publish(topic, Encoding.UTF8.GetBytes(jsonData), MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE, false);
        Console.WriteLine($"Message '{jsonData}' published to topic 'eKuiperTopic'");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error: {ex.Message}");
    }
}

```

# FILTER MIKROSERVIS

```
IConnection connection = new ConnectionFactory().CreateConnection("nats://nats:4222");
string sub1 = "dashboard";
string lastDate = "";
List<AirQualityData> airQualityList = new List<AirQualityData>();
1 reference
void Subscribe()
{

    string clientId = Guid.NewGuid().ToString();
    MqttClient mqttClient = new MqttClient("mosquitto", 1883, false, null, null, MqttSslProtocols.None);

    mqttClient.MqttMsgPublishReceived += Client_MqttMsgPublishReceived;

    try
    {
        if (!mqttClient.IsConnected)
        {
            mqttClient.Connect(clientId);
        }
        mqttClient.Subscribe(new string[] { "eKuiperTopic"}, new byte[] { MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE });
        Console.WriteLine($"Subscribed to topic 'eKuiperTopic'");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error: {ex.Message}");
    }
}
Subscribe();
```

```
Subscribe();
1 reference
void Client_MqttMsgPublishReceived(object sender, MqttMsgPublishEventArgs e)
{



        string message = Encoding.UTF8.GetString(e.Message);
        AirQualityData airQualityData = JsonConvert.DeserializeObject<AirQualityData>(message);
    if (lastDate == "")
    {
        lastDate = airQualityData.Date;
    }
    if (airQualityData.Date != lastDate)
        {
        sendToNatsTopic();
        lastDate = airQualityData.Date;
        airQualityList.Clear();
    }

       airQualityList.Add(airQualityData);



}
1 reference
void sendToNatsTopic()
{

    var aggregatedData = new
    {
        Date = airQualityList[0].DateTime.Date,
        Average_CO = airQualityList.Average(d => d.CO_GT),
        Average_PT08S1_CO = airQualityList.Average(d => d.PT08_S1_CO),
        Average_NMHC_GT = airQualityList.Average(d => d.NMHC_GT),
        Average_C6H6_GT = airQualityList.Average(d => d.C6H6_GT),
        Average_PT08S2_NMHC = airQualityList.Average(d => d.PT08_S2_NMHC),
        Average_NOx_GT = airQualityList.Average(d => d.NOx_GT),
        Average_PT08S3_NOx = airQualityList.Average(d => d.PT08_S3_NOx),
        Average_NO2_GT = airQualityList.Average(d => d.NO2_GT),
        Average_PT08S4_NO2 = airQualityList.Average(d => d.PT08_S4_NO2),
        Average_PT08S5_O3 = airQualityList.Average(d => d.PT08_S5_O3),
        Average_Temperature = airQualityList.Average(d => d.T),
        Average_RelativeHumidity = airQualityList.Average(d => d.RH),
        Average_AbsoluteHumidity = airQualityList.Average(d => d.AH)
    };

    connection.Publish(sub1, Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(aggregatedData)));

    Console.WriteLine(aggregatedData);
}
```

# DASHBOARD MIKROSERVIS

## Konfiguracioni fajlovi

```java
@Configuration  VVeljovic
public class NatsConfig {
  @Value("nats://nats:4222")
  private String natsUrl;


    @Bean  VVeljovic
    public Connection natsConnection() throws IOException, InterruptedException {
        Options options = new Options.Builder()
                .server(natsUrl).build();
        Connection natsConnection =  Nats.connect(options);
        return natsConnection;
    }
}
```

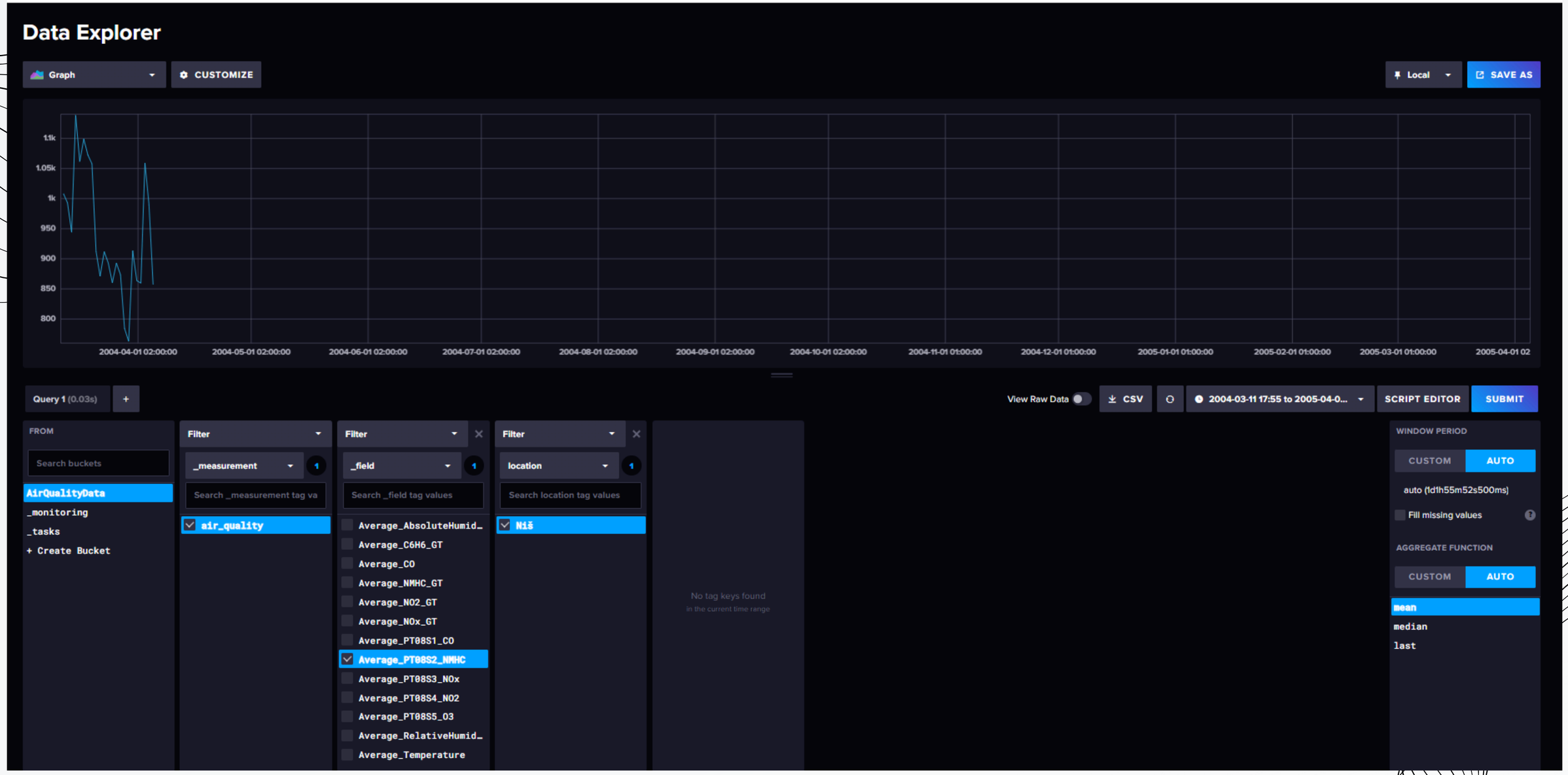
```java
@Configuration  VVeljovic
public class InfluxConfig {

    @Value("${influx.url}")
    private String influxUrl;

    @Value("${influx.username}")
    private String influxUserName;

    @Value("${influx.password}")
    private String influxPassword;

    @Value("${influx.token}")
    private String token;

    @Value("${influx.bucket}")
    private String bucket;

    @Value("${influx.org}")
    private String org;
    @Bean  VVeljovic
    public InfluxDBClient buildConnection()
    {
        return InfluxDBClientFactory.create(influxUrl,token.toCharArray(),org,bucket);
    }

}
```

# DASHBOARD MIKROSERVIS

```java
@Service  VVeljovic *
public class DashboardService {
    private NatsConfig natsConfig;  2 usages
    private InfluxConfig influxConfig;  2 usages
    private static final ObjectMapper mapper = new ObjectMapper();  1 usage
    private static final Logger logger = LoggerFactory.getLogger(DashboardApplication.class);  no usages
    @Autowired  VVeljovic
    public DashboardService(NatsConfig natsConfig, InfluxConfig influxConfig)  {
        this.natsConfig = natsConfig;
        this.influxConfig = influxConfig;
    }

    @EventListener(ApplicationReadyEvent.class)  VVeljovic *
    public void subscribeToNats() throws IOException, InterruptedException {

        Dispatcher dispatcher = natsConfig.natsConnection().createDispatcher();
        dispatcher.subscribe( subject: "dashboard", msg -> {
            String messageContent = new String(msg.getData(), StandardCharsets.UTF_8);
            try {
                AirQualityData airQualityData = mapper.readValue(messageContent, AirQualityData.class);
                System.out.println(airQualityData.getAverageCO());
                insertDataToInfluxDB(airQualityData);
            } catch (JsonProcessingException e) {
                throw new RuntimeException(e);
            } catch (ParseException e) {
                throw new RuntimeException(e);
            }
        });
    }
```

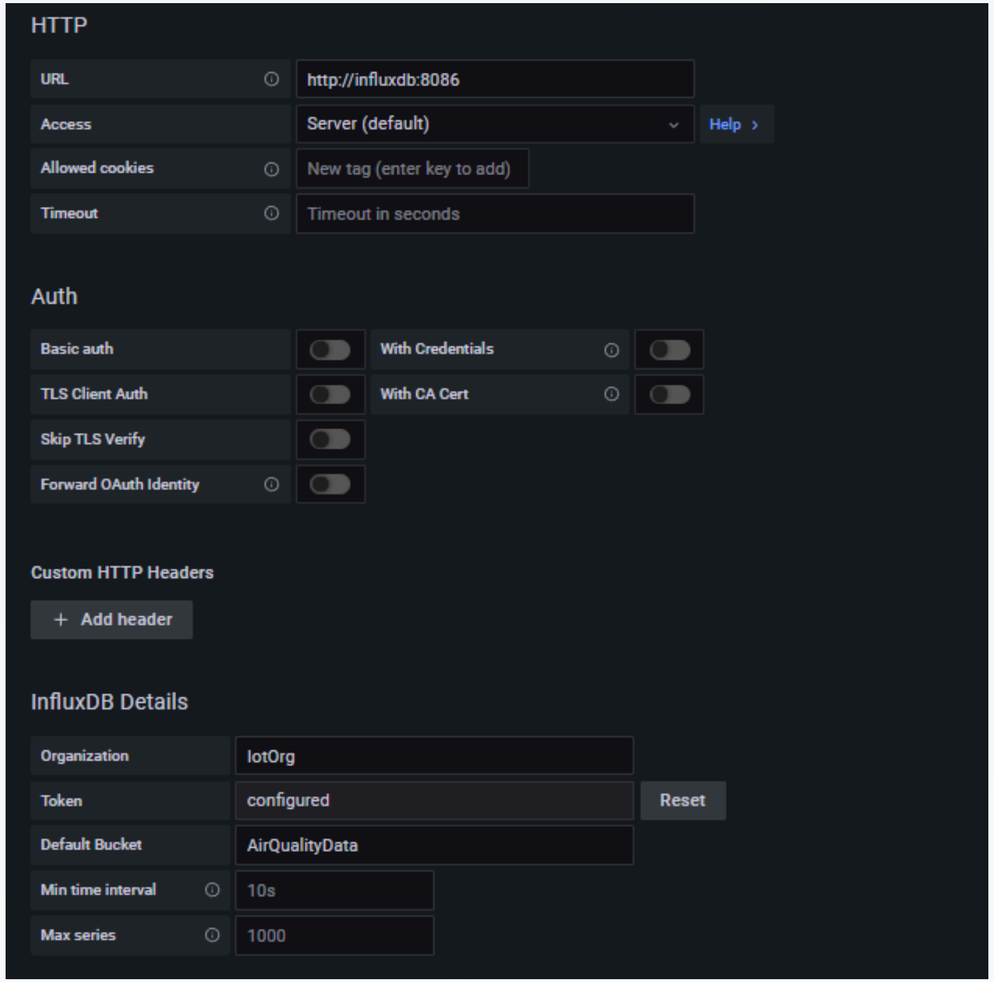
```java
public void insertDataToInfluxDB(AirQualityData airQualityData) throws ParseException {  1 usage  VVeljovic *
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd'T'HH:mm:ss");
    Instant timestamp = LocalDateTime.parse(airQualityData.getDate(), formatter) LocalDateTime
            .atOffset(ZoneOffset.UTC) OffsetDateTime
            .toInstant();
    Point point = Point.measurement( measurementName: "air_quality")
            .addTag("location", "Niš")
            .addField( field: "Average_CO", airQualityData.getAverageCO())
            .addField( field: "Average_PT08S1_CO", airQualityData.getAveragePT08S1CO())
            .addField( field: "Average_NMHC_GT", airQualityData.getAverageNMHCGT())
            .addField( field: "Average_C6H6_GT", airQualityData.getAverageC6H6GT())
            .addField( field: "Average_PT08S2_NMHC", airQualityData.getAveragePT08S2NMHC())
            .addField( field: "Average_NOx_GT", airQualityData.getAverageNOxGT())
            .addField( field: "Average_PT08S3_NOx", airQualityData.getAveragePT08S3NOx())
            .addField( field: "Average_NO2_GT", airQualityData.getAverageNO2GT())
            .addField( field: "Average_PT08S4_NO2", airQualityData.getAveragePT08S4NO2())
            .addField( field: "Average_PT08S5_O3", airQualityData.getAveragePT08S5O3())
            .addField( field: "Average_Temperature", airQualityData.getAverageTemperature())
            .addField( field: "Average_RelativeHumidity", airQualityData.getAverageRelativeHumidity())
            .addField( field: "Average_AbsoluteHumidity", airQualityData.getAverageAbsoluteHumidity())
            .time(timestamp, WritePrecision.MS);
            System.out.println(point);
    influxConfig.buildConnection().getWriteApiBlocking().writePoint(point);

}
```

# INFLUXDB

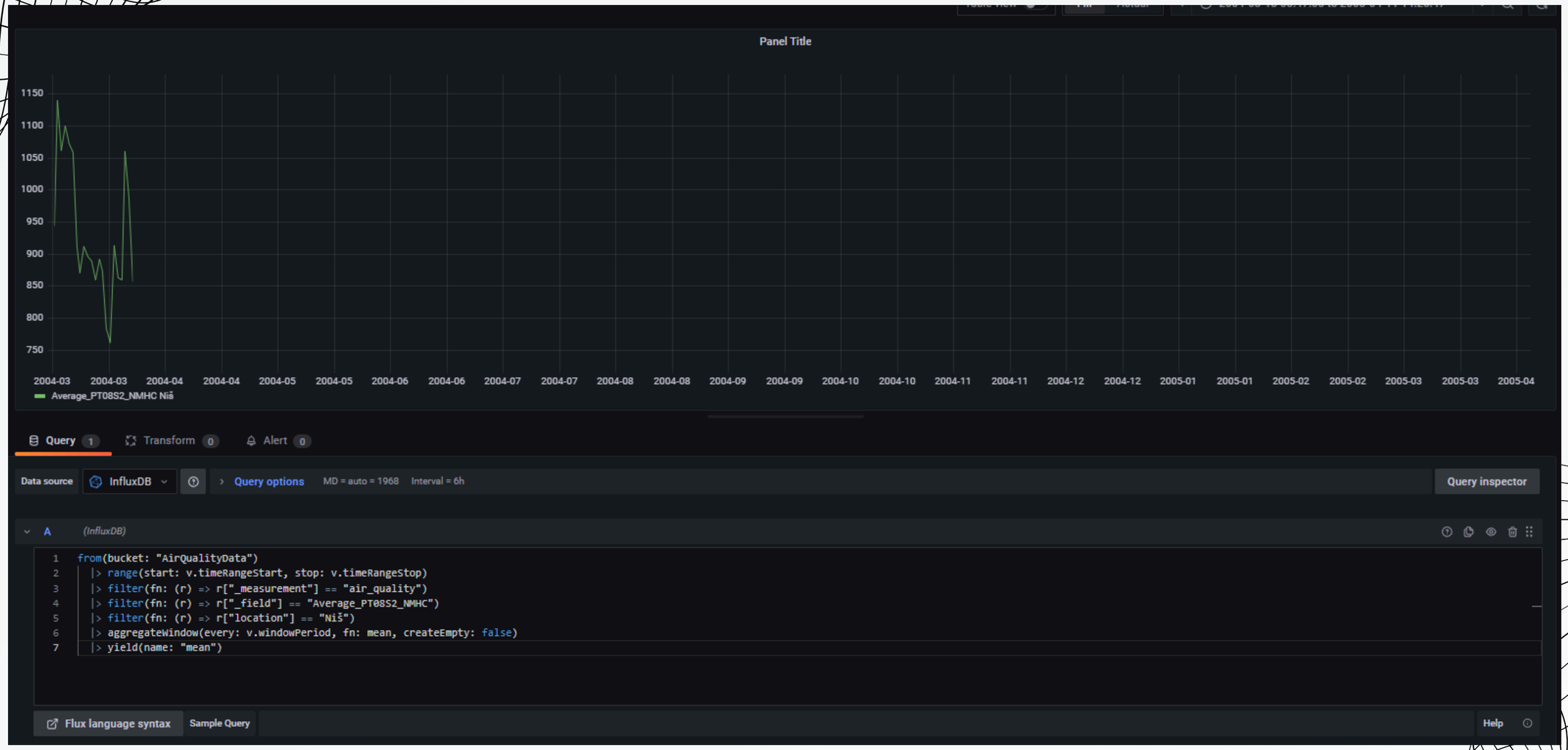

Data Explorer
Graph
CUSTOMIZE
Local
SAVE AS
1.1k
1.05k
1k
950
900
850
800
2004-04-01 02:00:00
2004-05-01 02:00:00
2004-06-01 02:00:00
2004-07-01 02:00:00
2004-08-01 02:00:00
2004-09-01 02:00:00
2004-10-01 02:00:00
2004-11-01 01:00:00
2004-12-01 01:00:00
2005-01-01 01:00:00
2005-02-01 01:00:00
2005-03-01 01:00:00
2005-04-01 02
Query 1 (0.03s)
+
View Raw Data
CSV
2004-03-11 17:55 to 2005-04-0...
SCRIPT EDITOR
SUBMIT
FROM
Search buckets
AirQualityData
_monitoring
_tasks
+ Create Bucket
Filter
_measurement
Search _measurement tag va
air_quality
Filter
_field
Search _field tag values
Average_AbsoluteHumid_
Average_C6H6_GT
Average_CO
Average_NMHC_GT
Average_NO2_GT
Average_NOx_GT
Average_PT08S1_CO
Average_PT08S2_NMHC
Average_PT08S3_NOx
Average_PT08S4_NO2
Average_PT08S5_O3
Average_RelativeHumid_
Average_Temperature
Filter
location
Search location tag values
Niš
No tag keys found
in the current time range
WINDOW PERIOD
CUSTOM
AUTO
auto (1d1h55m52s500ms)
Fill missing values
AGGREGATE FUNCTION
CUSTOM
AUTO
mean
median
last

# GRAFANA

## Add data source



# GRAFANA


Panel Title
1150
1100
1050
1000
950
900
850
800
750
2004-03
2004-03
2004-04
2004-04
2004-05
2004-05
2004-06
2004-06
2004-07
2004-07
2004-08
2004-08
2004-09
2004-09
2004-10
2004-10
2004-11
2004-11
2004-12
2004-12
2005-01
2005-01
2005-02
2005-02
2005-03
2005-03
2005-04
Average_PT08S2_NMHC Niš
Query 1
Transform 0
Alert 0
Data source
InfluxDB
Query options
MD = auto = 1968
Interval = 6h
Query inspector
A
(InfluxDB)
from(bucket: "AirQualityData")
|> range(start: v.timeRangeStart, stop: v.timeRangeStop)
|> filter(fn: (r) => r["_measurement"] == "air_quality")
|> filter(fn: (r) => r["_field"] == "Average_PT08S2_NMHC")
|> filter(fn: (r) => r["location"] == "Niš")
|> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
|> yield(name: "mean")
Flux language syntax
Sample Query
Help

# COMMAND MIKROSERVIS

```typescript
@WebSocketGateway({
  cors: {
    origin: 'http://127.0.0.1:5500',
    methods: ['GET', 'POST'],
    credentials: true,
  }
})
export class WebsocketGateway implements OnGatewayInit, OnGatewayConnection, OnGatewayDisconnect {
  private clients: Set<Socket> = new Set();

  @WebSocketServer() server: Server;

  afterInit(server: Server) {
      console.log('WebSocket Gateway initialized');
  }

  handleConnection(client: Socket) {
      console.log(`Client connected: ${client.id}`);
      this.clients.add(client);
  }

  handleDisconnect(client: Socket) {
      console.log(`Client disconnected: ${client.id}`);
      this.clients.delete(client);
  }

  @SubscribeMessage('message')
  handleMessage(client: Socket, payload: any): void {
      console.log(`Message from client ${client.id}: ${payload}`);
      this.server.emit('messageToClient', payload);
  }
}
```

```typescript
@Injectable()
export class MqttService {
  public readonly mqtt: MqttClient;
  private readonly PT08_S1_CO: number = 1500;
  private readonly PT08_S2_NMHC: number = 1400;
  private readonly PT08_S3_NOx: number = 1700;
  private readonly PT08_S4_NO2: number = 1700;
  private readonly PT08_S5_O3: number = 1900;
  constructor(private readonly webSocket: WebsocketGateway) {
    this.mqtt = connect('http://mosquitto:1883', {
      clientId: 'nestClient',
      clean: true,
      connectTimeout: parseInt('4000', 10),
      reconnectPeriod: parseInt('1000', 10),
    });
    this.mqtt.on('connect', () => {
      console.log('Connected to MQTT server');
    });
    this.mqtt.subscribe('resultTopic', { qos: 1 });
this.mqtt.on('message', (topic, message) => {
    const jsonString = message.toString();
    console.log(jsonString);

    try {
      const mqttData: MqttData = JSON.parse(jsonString);
      console.log('Parsed MQTT Data:', mqttData);


      let invalidParametersMessage = '';

      if (mqttData.PT08_S1_CO > this.PT08_S1_CO) {
        const increasePercentage = ((mqttData.PT08_S1_CO - this.PT08_S1_CO) / this.PT08_S1_CO) * 100;
        invalidParametersMessage += `Increase in PT08_S1_CO is ${increasePercentage.toFixed(2)}%\n`;
      }

      if (mqttData.PT08_S2_NMHC > this.PT08_S2_NMHC) {
        const increasePercentage = ((mqttData.PT08_S2_NMHC - this.PT08_S2_NMHC) / this.PT08_S2_NMHC) * 100;
        invalidParametersMessage += `Increase in PT08_S2_NMHC is ${increasePercentage.toFixed(2)}%\n`;
      }
```

```typescript
  if (invalidParametersMessage !== '') {
    webSocket.server.emit('messageToClient', jsonString);
    console.log('Invalid parameters detected:');
    console.log(invalidParametersMessage);


    webSocket.server.emit('messageToClient', invalidParametersMessage);
    webSocket.server.emit('messageToClient', "----------------------------------------------------------------");
}
```

# HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>AirQuality data</title>
</head>
<body>
  <h1>AirQuality data</h1>
  <div id="messages"></div>

  <script src="https://cdn.socket.io/4.0.0/socket.io.min.js"></script>
  <script>
    const socket = io('http://localhost:3001');

    socket.on('messageToClient', (message) => {
      const messagesDiv = document.getElementById('messages');
      const messageElement = document.createElement('div');
      messageElement.innerText = message;
      messagesDiv.appendChild(messageElement);
    });


  </script>
</body>
</html>
```

**AirQuality data**

{"AH":1.1123,"C6H6_GT":16.5,"CO_GT":-200,"Date":"19/04/2004","NMHC_GT":-200,"NO2_GT":-200,"NOx_GT":-200,"PT08_S1_CO":1399,"PT08_S2_NMHC":1195,"PT08_S3_NOx":680,"PT08_S4_NO2":1976,"PT08_S5_O3":1308,"RH":64.2,"T":15.4,"Time":"07:00:00"}
Increase in PT08_S4_NO2 is 16.24%
---
{"AH":1.0826,"C6H6_GT":18.7,"CO_GT":-200,"Date":"19/04/2004","NMHC_GT":-200,"NO2_GT":-200,"NOx_GT":-200,"PT08_S1_CO":1325,"PT08_S2_NMHC":1260,"PT08_S3_NOx":652,"PT08_S4_NO2":1969,"PT08_S5_O3":1303,"RH":57.9,"T":16.5,"Time":"08:00:00"}
Increase in PT08_S4_NO2 is 15.82%
---
{"AH":0.9569,"C6H6_GT":10.9,"CO_GT":-200,"Date":"19/04/2004","NMHC_GT":-200,"NO2_GT":-200,"NOx_GT":-200,"PT08_S1_CO":1105,"PT08_S2_NMHC":1010,"PT08_S3_NOx":811,"PT08_S4_NO2":1709,"PT08_S5_O3":947,"RH":46.7,"T":18,"Time":"12:00:00"}
Increase in PT08_S4_NO2 is 0.53%
---
{"AH":1.0813,"C6H6_GT":13.4,"CO_GT":-200,"Date":"19/04/2004","NMHC_GT":-200,"NO2_GT":-200,"NOx_GT":-200,"PT08_S1_CO":1194,"PT08_S2_NMHC":1097,"PT08_S3_NOx":743,"PT08_S4_NO2":1847,"PT08_S5_O3":1049,"RH":58.2,"T":16.5,"Time":"13:00:00"}
Increase in PT08_S4_NO2 is 8.65%
---
{"AH":1.0016,"C6H6_GT":11.5,"CO_GT":-200,"Date":"19/04/2004","NMHC_GT":-200,"NO2_GT":-200,"NOx_GT":-200,"PT08_S1_CO":1207,"PT08_S2_NMHC":1032,"PT08_S3_NOx":797,"PT08_S4_NO2":1701,"PT08_S5_O3":969,"RH":79,"T":10.5,"Time":"16:00:00"}
Increase in PT08_S4_NO2 is 0.06%
---
{"AH":1.0135,"C6H6_GT":20.7,"CO_GT":-200,"Date":"20/04/2004","NMHC_GT":-200,"NO2_GT":-200,"NOx_GT":-200,"PT08_S1_CO":1430,"PT08_S2_NMHC":1316,"PT08_S3_NOx":638,"PT08_S4_NO2":2069,"PT08_S5_O3":1382,"RH":77.4,"T":11,"Time":"07:00:00"}
Increase in PT08_S4_NO2 is 21.71%
---
{"AH":0.9906,"C6H6_GT":29.9,"CO_GT":-200,"Date":"20/04/2004","NMHC_GT":-200,"NO2_GT":-200,"NOx_GT":-200,"PT08_S1_CO":1565,"PT08_S2_NMHC":1549,"PT08_S3_NOx":540,"PT08_S4_NO2":2396,"PT08_S5_O3":1645,"RH":69.2,"T":12.3,"Time":"08:00:00"}
Increase in PT08_S1_CO is 4.33%
Increase in PT08_S2_NMHC is 10.64%
Increase in PT08_S4_NO2 is 40.94%
---
{"AH":0.9936,"C6H6_GT":20.1,"CO_GT":-200,"Date":"20/04/2004","NMHC_GT":-200,"NO2_GT":-200,"NOx_GT":-200,"PT08_S1_CO":1365,"PT08_S2_NMHC":1299,"PT08_S3_NOx":637,"PT08_S4_NO2":1999,"PT08_S5_O3":1514,"RH":60.4,"T":14.5,"Time":"09:00:00"}
Increase in PT08_S4_NO2 is 17.59%
---
{"AH":1.0185,"C6H6_GT":17.1,"CO_GT":-200,"Date":"20/04/2004","NMHC_GT":-200,"NO2_GT":-200,"NOx_GT":-200,"PT08_S1_CO":1348,"PT08_S2_NMHC":1214,"PT08_S3_NOx":678,"PT08_S4_NO2":1874,"PT08_S5_O3":1509,"RH":56.3,"T":16,"Time":"10:00:00"}
Increase in PT08_S4_NO2 is 10.24%
---

# HVALA NA PAŽNJI

LARANA, INC.