Projekat 2 – IoT

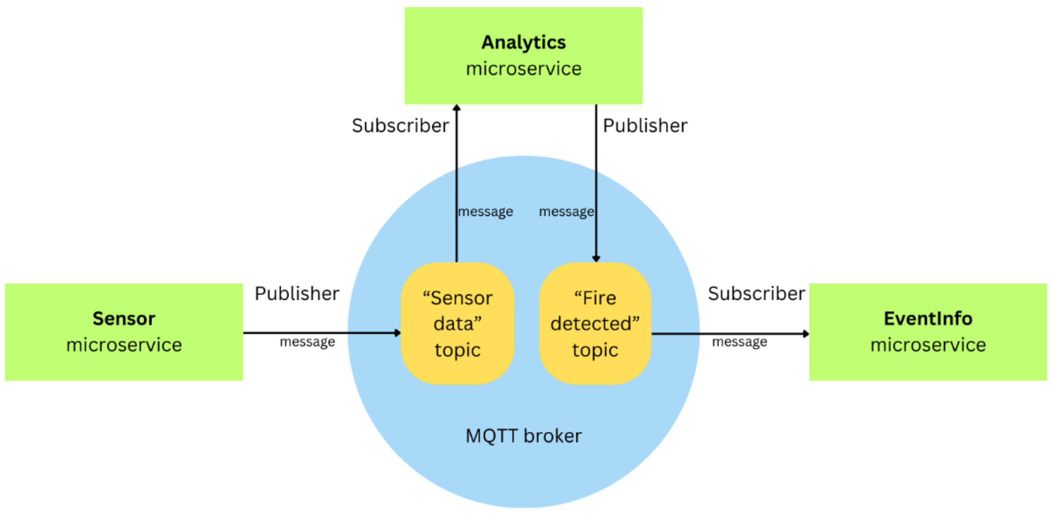
Zahtevi

Cilj ovog projekta je implementirati 3 mikroservisa (Sensor, Analytics i EventInfo) u 3 različite tehnologije (.NET, Python/Flask i NestJS) radi simulacije očitavanja podataka sa senzora radi analize i obezbedjivanja odgovarajućeg dešavanja na osnovu rezultata analize. Za komunikaciju se koristi MQTT message broker, a svaki mikroservis je startovan kao Docker container.

Odabrane tehnologije

Mikroservis **Sensor**	.NET
Mikroservis **Analytics**	Python/Flask
Mikroservis **EventInfo**	NodeJS
MQTT broker	Mosquitto
Baza podataka	MongoDB

Arhitektura



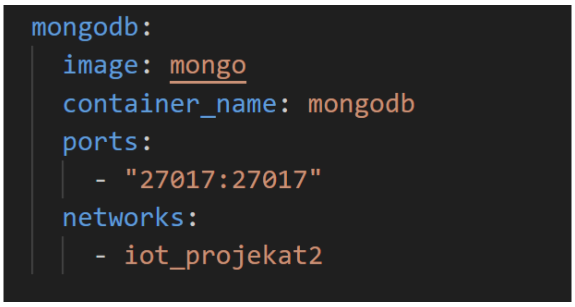
Baza podataka - Iskorišćena je baza podataka iz prvog projekta i smeštena je u container **mongodb**, koji se popunjava na osnovu insertData.py skripte koja je pokrenuta u container-u **python-script** tako da se popunjavanje vrši pri pokretanju *Docker-compose* file-a.

Mikroservis **Sensor** - Povezuje se sa bazom i čita podatke koje objavljuje na topic *„Sensor data"* MQTT brokera, simulirajući očitavanje podataka sa realnih senzora.

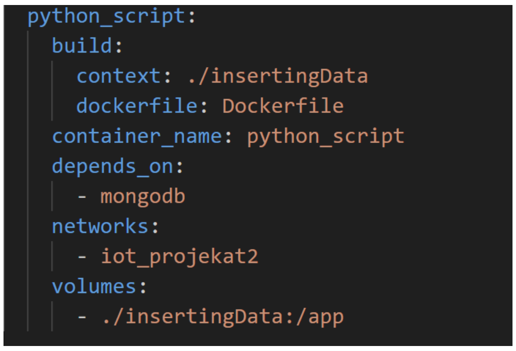
Mikroservis **Analytics** – Pretplaćuje se na topic *„Sensor data"* MQTT brokera, tako da dobija podatke koji su obajvljeni na ovom topic-u. Ako u primljenim podacima registruje vrednost gde je protivpožarni alarm potrebno aktivirati, objavljuje te podatke na topic „Fire detected" MQTT brokera.

Mikroservis **EventInfo** - Pretplaćuje se na topic „Fire detected“ MQTT brokera, tako da dobija podatke o vrednostima sa senzora koje su izazvale vatru i obezbedjuje prikaz poslednjih vrednosti preko REST API-ja.

Kreiranje mongodb container-a i popunjavanje baze podacima



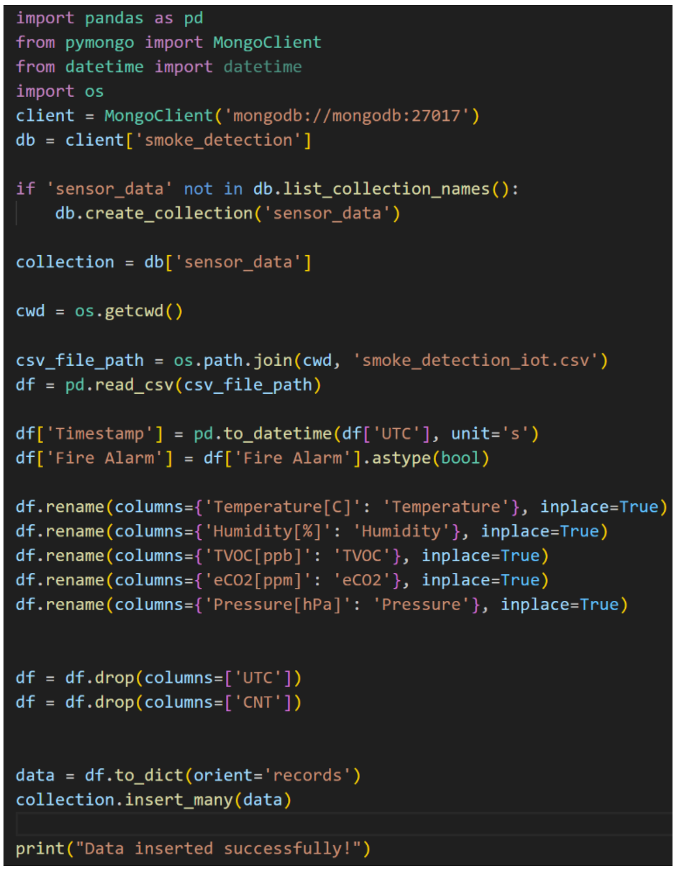
```yaml
mongodb:
  image: mongo
  container_name: mongodb
  ports:
    - "27017:27017"
  networks:
    - iot_projekat2
```



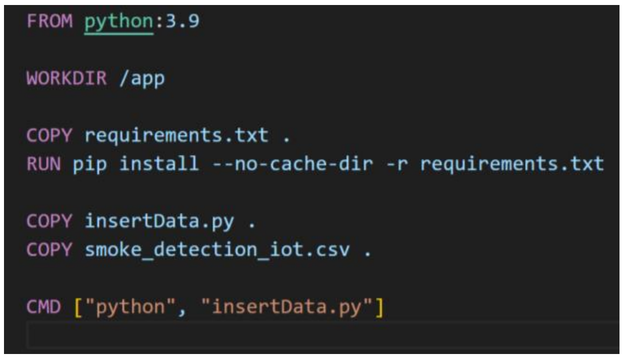
```yaml
python_script:
  build:
    context: ./insertingData
    dockerfile: Dockerfile
  container_name: python_script
  depends_on:
    - mongodb
  networks:
    - iot_projekat2
  volumes:
    - ./insertingData:/app
```

Deo koda iz *Docker-compose.yaml* fajla.

Skripta za popunjavanje baze podataka se pokreće u okviru python_script containera-a.



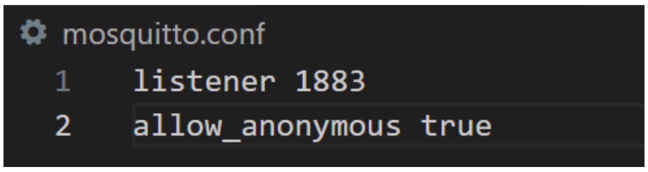
```python
import pandas as pd
from pymongo import MongoClient
from datetime import datetime
import os
client = MongoClient('mongodb://mongodb:27017')
db = client['smoke_detection']

if 'sensor_data' not in db.list_collection_names():
    db.create_collection('sensor_data')

collection = db['sensor_data']

cwd = os.getcwd()

csv_file_path = os.path.join(cwd, 'smoke_detection_iot.csv')
df = pd.read_csv(csv_file_path)

df['Timestamp'] = pd.to_datetime(df['UTC'], unit='s')
df['Fire Alarm'] = df['Fire Alarm'].astype(bool)

df.rename(columns={'Temperature[C]': 'Temperature'}, inplace=True)
df.rename(columns={'Humidity[%]': 'Humidity'}, inplace=True)
df.rename(columns={'TVOC[ppb]': 'TVOC'}, inplace=True)
df.rename(columns={'eCO2[ppm]': 'eCO2'}, inplace=True)
df.rename(columns={'Pressure[hPa]': 'Pressure'}, inplace=True)


df = df.drop(columns=['UTC'])
df = df.drop(columns=['CNT'])


data = df.to_dict(orient='records')
collection.insert_many(data)

print("Data inserted successfully!")
```



```dockerfile
FROM python:3.9

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY insertData.py .
COPY smoke_detection_iot.csv .

CMD ["python", "insertData.py"]
```

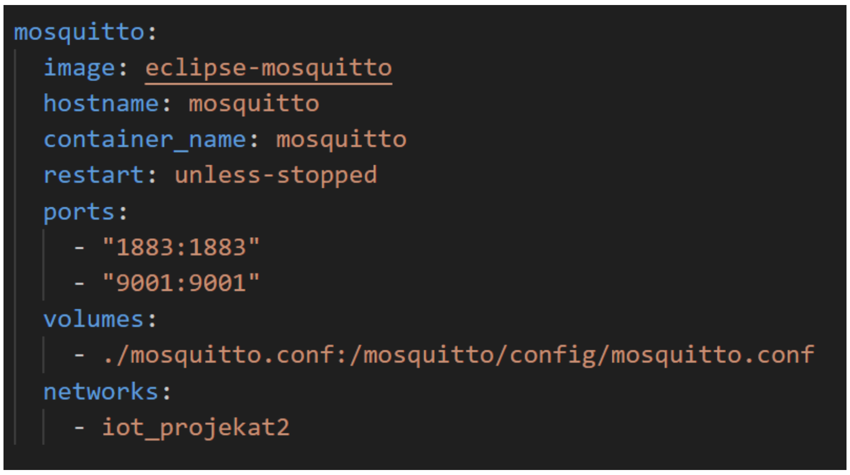
Dockerfile potreban za pokretanje insertData.py skripte.

Kreiranje MQTT brokera



mosquitto.conf
```
listener 1883
allow_anonymous true
```

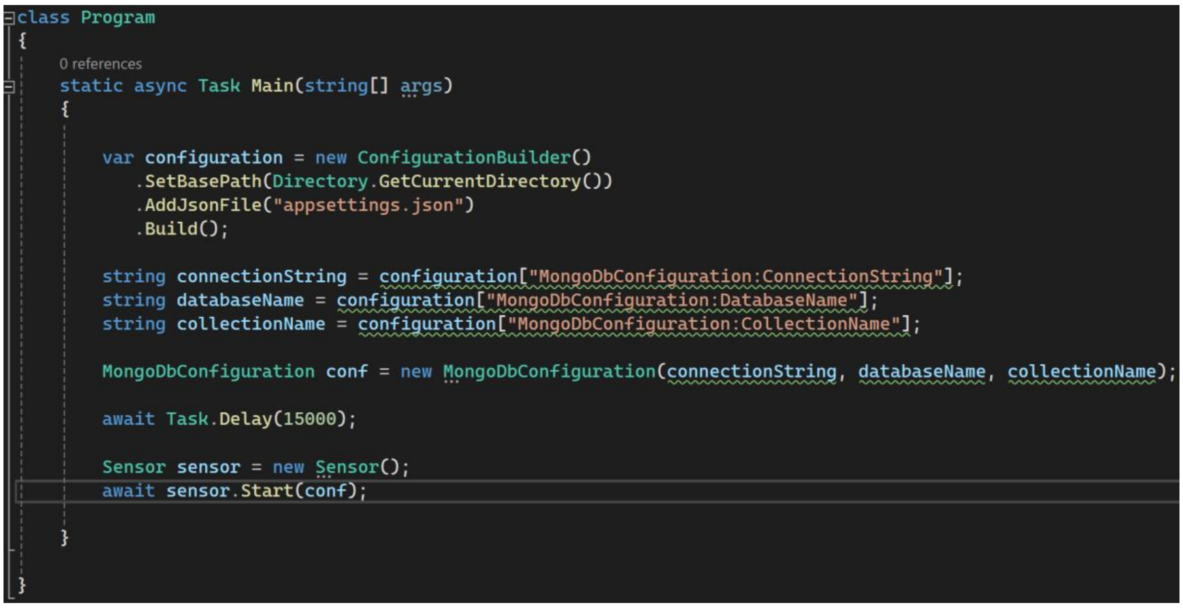
Neophodno je definisati *mosquitto.conf* fajl kako bi bilo moguće kreiranje container-a u kome se pokreće MQTT broker.



```yaml
mosquitto:
  image: eclipse-mosquitto
  hostname: mosquitto
  container_name: mosquitto
  restart: unless-stopped
  ports:
    - "1883:1883"
    - "9001:9001"
  volumes:
    - ./mosquitto.conf:/mosquitto/config/mosquitto.conf
  networks:
    - iot_projekat2
```

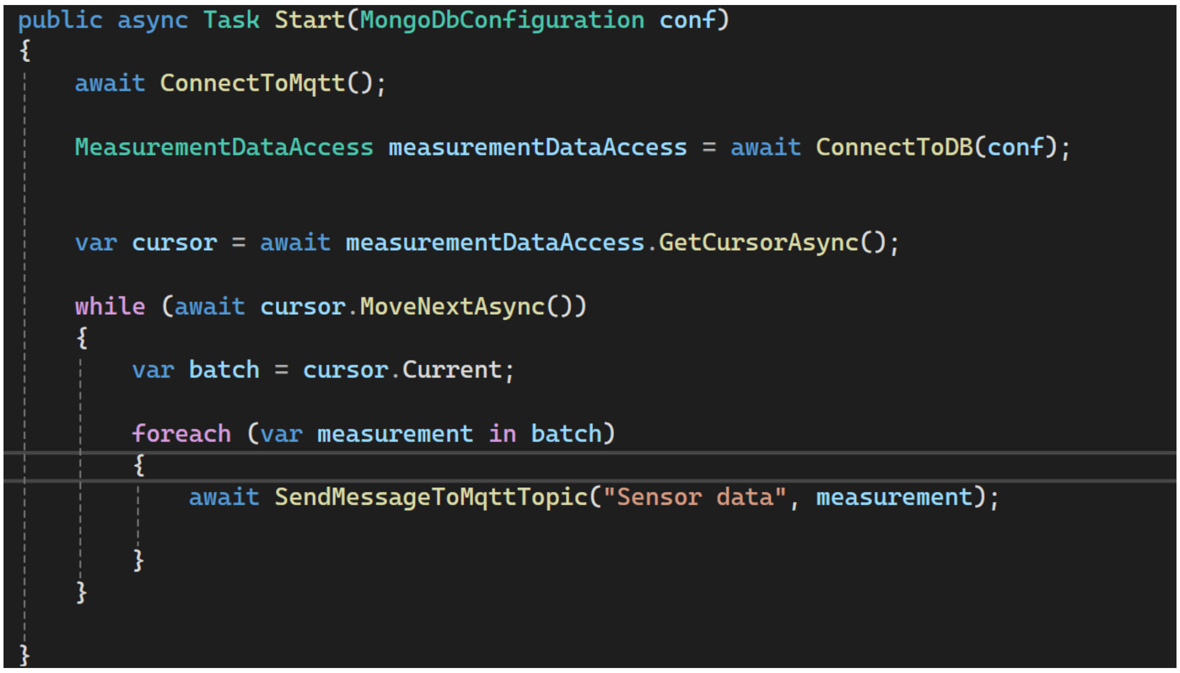
Deo *Docker-compose.yaml* fajla pomoću kojeg se pokreće MQTT broker u container-u koji koristi *eclipse-mosquitto* image.

Kreiranje Sensor mikroservisa



```csharp
class Program
{
    0 references
    static async Task Main(string[] args)
    {

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("appsettings.json")
            .Build();

        string connectionString = configuration["MongoDbConfiguration:ConnectionString"];
        string databaseName = configuration["MongoDbConfiguration:DatabaseName"];
        string collectionName = configuration["MongoDbConfiguration:CollectionName"];

        MongoDbConfiguration conf = new MongoDbConfiguration(connectionString, databaseName, collectionName);

        await Task.Delay(15000);

        Sensor sensor = new Sensor();
        await sensor.Start(conf);

    }

}
```

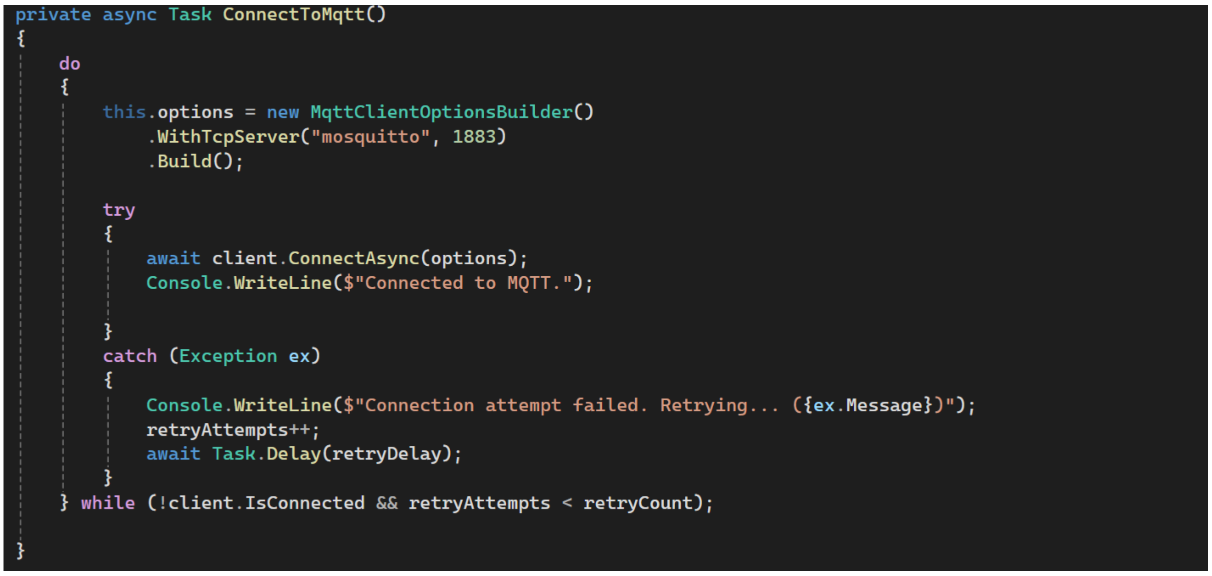
Podaci potrebni za konfigurisanje konekcije sa bazom podataka se čuvaju u *appsettings.json* fajlu i prilikom pokretanja se prikupljaju i prosledjuju *Sensor* mikroservisu pri njegovom pokretanju.



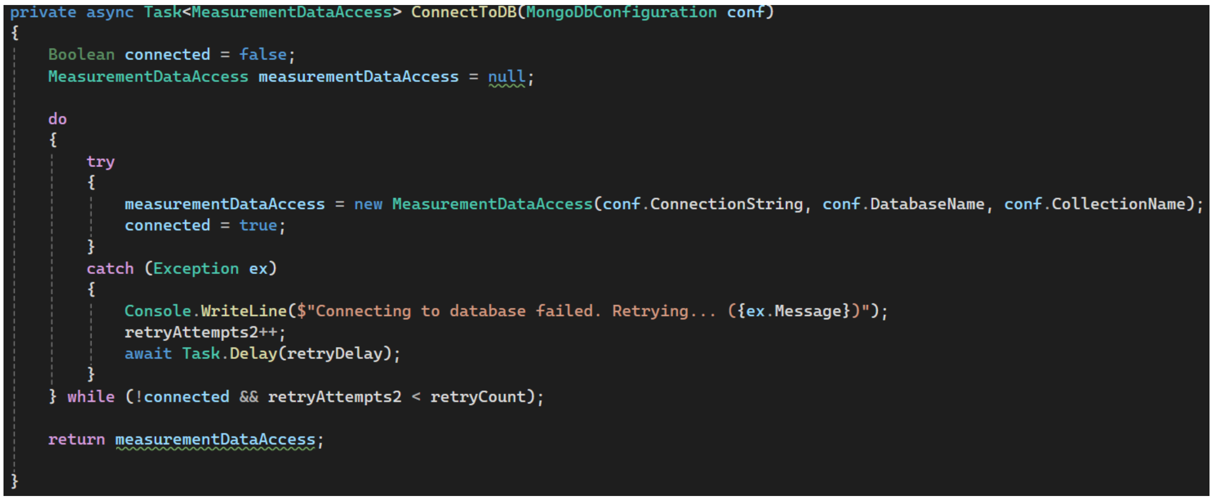
```
public async Task Start(MongoDbConfiguration conf)
{
    await ConnectToMqtt();

    MeasurementDataAccess measurementDataAccess = await ConnectToDB(conf);


    var cursor = await measurementDataAccess.GetCursorAsync();

    while (await cursor.MoveNextAsync())
    {
        var batch = cursor.Current;

        foreach (var measurement in batch)
        {
            await SendMessageToMqttTopic("Sensor data", measurement);

        }
    }

}
```

Prilikom startovanja potrebno je konektovati se sa MQTT brokerom, a zatim i sa bazom. Pročitani podaci se šalju na *„Sensor data"* topic.

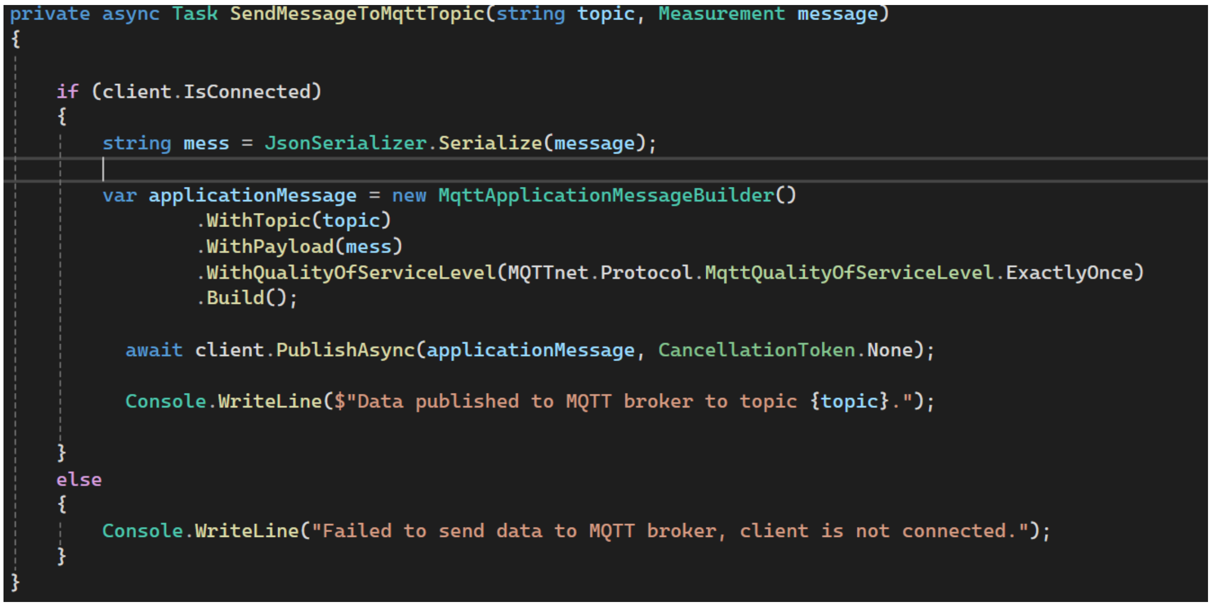
U nastavku slede funkcije koje obezbedjuju konekciju sa bazom, brokerom i funkcija za slanje poruka na odgovarajući topic.



```
private async Task ConnectToMqtt()
{
    do
    {
        this.options = new MqttClientOptionsBuilder()
            .WithTcpServer("mosquitto", 1883)
            .Build();

        try
        {
            await client.ConnectAsync(options);
            Console.WriteLine($"Connected to MQTT.");

        }
        catch (Exception ex)
        {
            Console.WriteLine($"Connection attempt failed. Retrying... ({ex.Message})");
            retryAttempts++;
            await Task.Delay(retryDelay);
        }
    } while (!client.IsConnected && retryAttempts < retryCount);

}
```



```csharp
private async Task<MeasurementDataAccess> ConnectToDB(MongoDbConfiguration conf)
{
    Boolean connected = false;
    MeasurementDataAccess measurementDataAccess = null;

    do
    {
        try
        {
            measurementDataAccess = new MeasurementDataAccess(conf.ConnectionString, conf.DatabaseName, conf.CollectionName);
            connected = true;
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Connecting to database failed. Retrying... ({ex.Message})");
            retryAttempts2++;
            await Task.Delay(retryDelay);
        }
    } while (!connected && retryAttempts2 < retryCount);

    return measurementDataAccess;
}
```

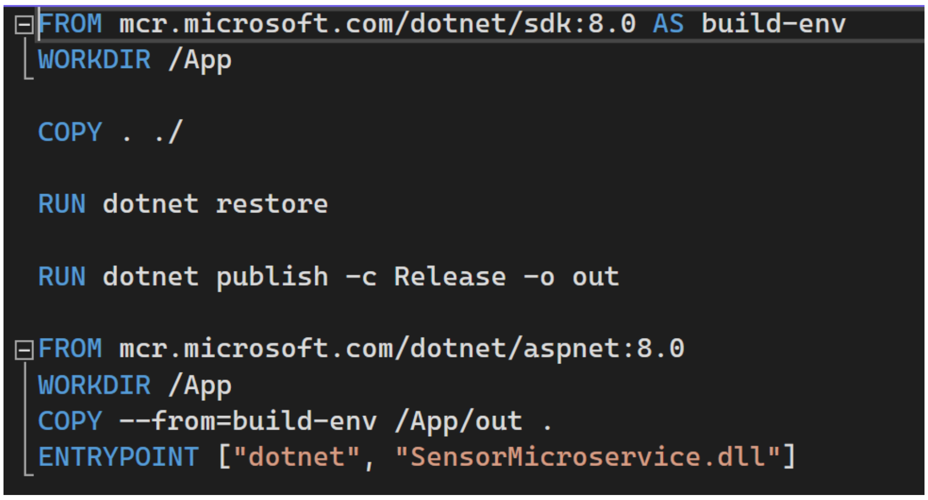


```csharp
private async Task SendMessageToMqttTopic(string topic, Measurement message)
{

    if (client.IsConnected)
    {
        string mess = JsonSerializer.Serialize(message);

        var applicationMessage = new MqttApplicationMessageBuilder()
                .WithTopic(topic)
                .WithPayload(mess)
                .WithQualityOfServiceLevel(MQTTnet.Protocol.MqttQualityOfServiceLevel.ExactlyOnce)
                .Build();

         await client.PublishAsync(applicationMessage, CancellationToken.None);

         Console.WriteLine($"Data published to MQTT broker to topic {topic}.");

    }
    else
    {
        Console.WriteLine("Failed to send data to MQTT broker, client is not connected.");
    }
}
```

Naredba za kreiranje image-a za ovaj mikroservis:

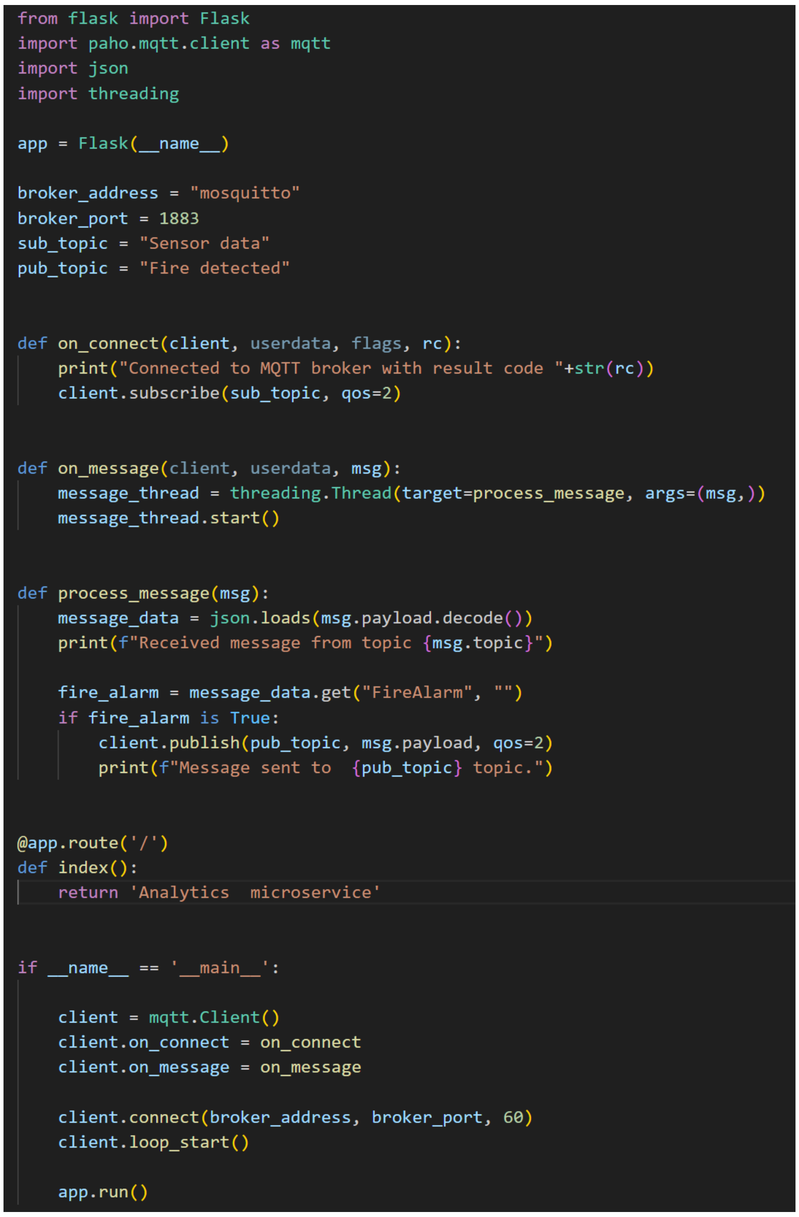
```
docker build -t sensor_ms -f Dockerfile .
```

Dockerfile:



```dockerfile
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build-env
WORKDIR /App

COPY . ./

RUN dotnet restore

RUN dotnet publish -c Release -o out

FROM mcr.microsoft.com/dotnet/aspnet:8.0
WORKDIR /App
COPY --from=build-env /App/out .
ENTRYPOINT ["dotnet", "SensorMicroservice.dll"]
```

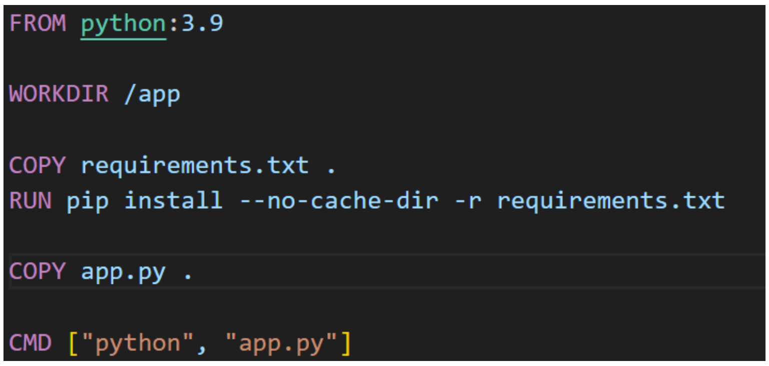
Kreiranje **Analytics** mikroservisa



```python
from flask import Flask
import paho.mqtt.client as mqtt
import json
import threading

app = Flask(__name__)

broker_address = "mosquitto"
broker_port = 1883
sub_topic = "Sensor data"
pub_topic = "Fire detected"


def on_connect(client, userdata, flags, rc):
    print("Connected to MQTT broker with result code "+str(rc))
    client.subscribe(sub_topic, qos=2)


def on_message(client, userdata, msg):
    message_thread = threading.Thread(target=process_message, args=(msg,))
    message_thread.start()


def process_message(msg):
    message_data = json.loads(msg.payload.decode())
    print(f"Received message from topic {msg.topic}")

    fire_alarm = message_data.get("FireAlarm", "")
    if fire_alarm is True:
        client.publish(pub_topic, msg.payload, qos=2)
        print(f"Message sent to  {pub_topic} topic.")


@app.route('/')
def index():
    return 'Analytics  microservice'


if __name__ == '__main__':

    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message

    client.connect(broker_address, broker_port, 60)
    client.loop_start()

    app.run()
```

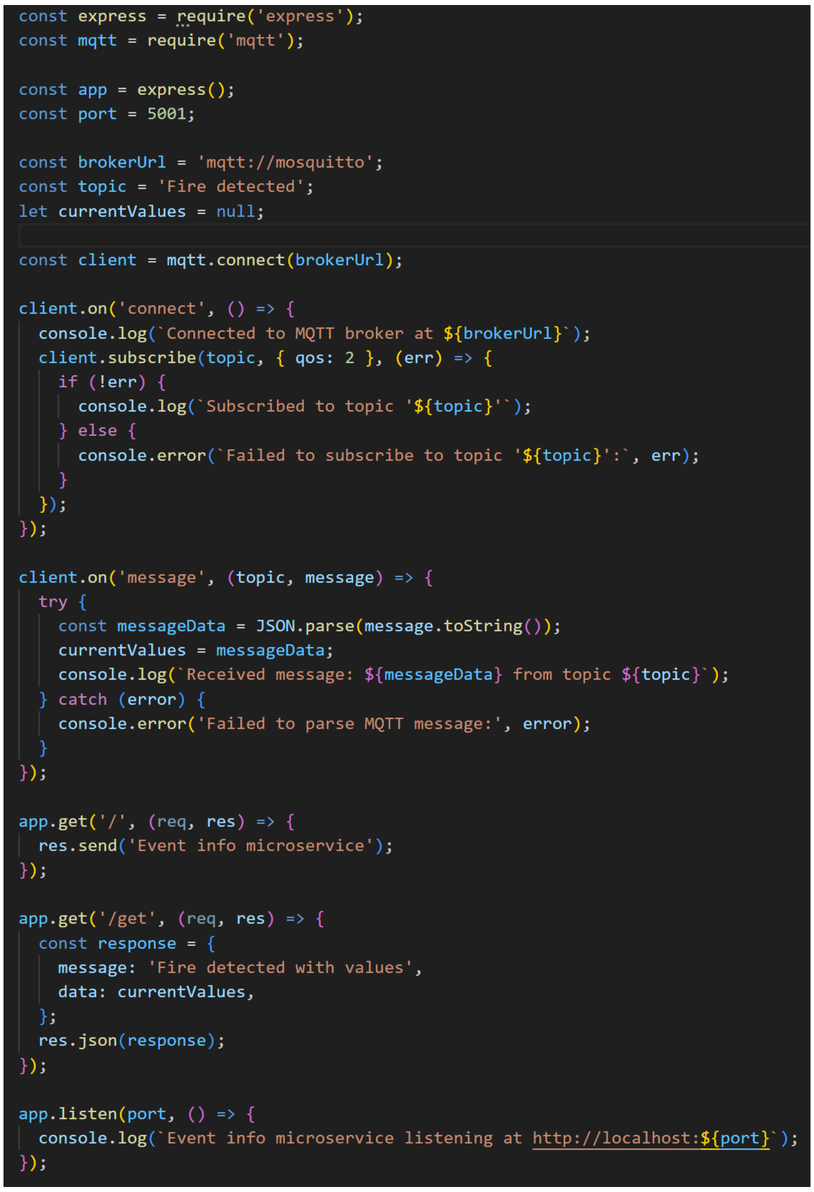
Kreira se MQTT klijent koji se pretplaćuje na *sub_topic* i analizira svaku poruku koju dobije u posebnoj niti. U slučaju detekcije vrednosti koje ukazuju na prisustvo vatre šalje poruku na *pub_topic.*

Dockerfile:



```dockerfile
FROM python:3.9

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY app.py .

CMD ["python", "app.py"]
```

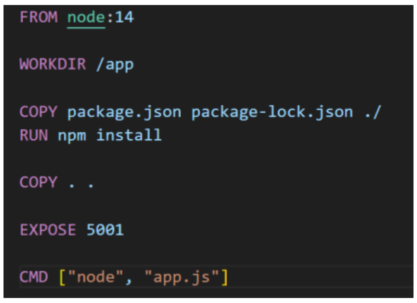
Kreiranje **EventInfo** mikroservisa



```javascript
const express = require('express');
const mqtt = require('mqtt');

const app = express();
const port = 5001;

const brokerUrl = 'mqtt://mosquitto';
const topic = 'Fire detected';
let currentValues = null;

const client = mqtt.connect(brokerUrl);

client.on('connect', () => {
  console.log(`Connected to MQTT broker at ${brokerUrl}`);
  client.subscribe(topic, { qos: 2 }, (err) => {
    if (!err) {
      console.log(`Subscribed to topic '${topic}'`);
    } else {
      console.error(`Failed to subscribe to topic '${topic}':`, err);
    }
  });
});

client.on('message', (topic, message) => {
  try {
    const messageData = JSON.parse(message.toString());
    currentValues = messageData;
    console.log(`Received message: ${messageData} from topic ${topic}`);
  } catch (error) {
    console.error('Failed to parse MQTT message:', error);
  }
});

app.get('/', (req, res) => {
  res.send('Event info microservice');
});

app.get('/get', (req, res) => {
  const response = {
    message: 'Fire detected with values',
    data: currentValues,
  };
  res.json(response);
});

app.listen(port, () => {
  console.log(`Event info microservice listening at http://localhost:${port}`);
});
```

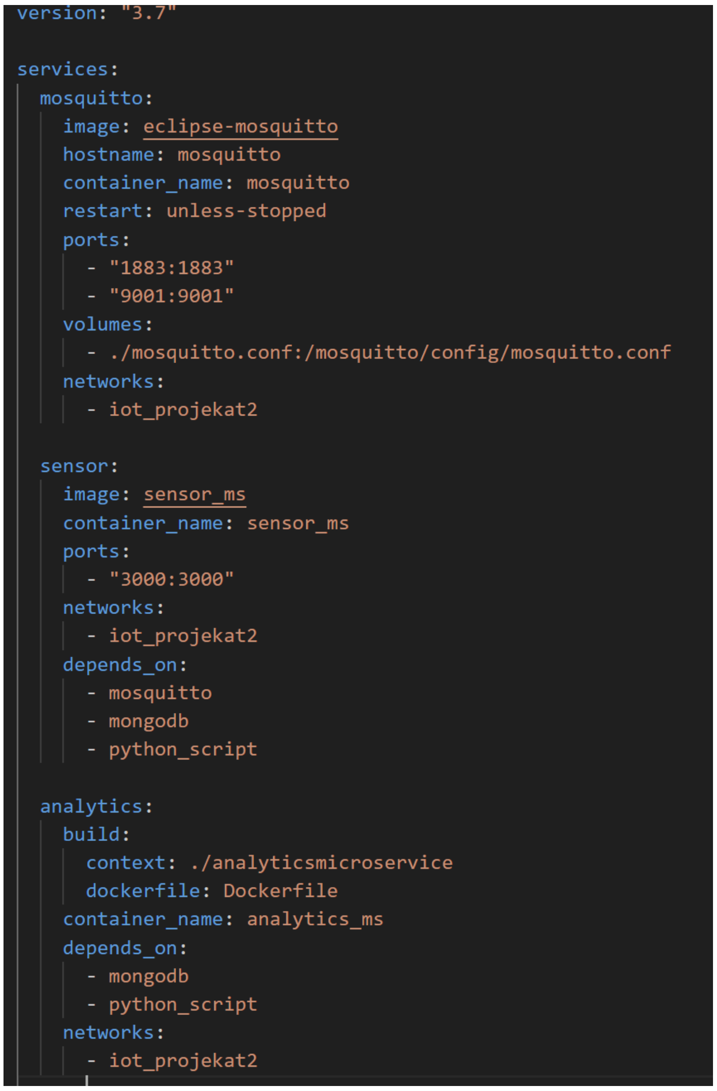
Kreira se MQTT klijent koji se pretplaćuje na *topic* i pamti vrednosti iz poruke u lokalnu promenljivu koja čuva poslednju vrednost koja je dovela do detektovanja vatre. Ovaj servis obezbedjuje REST API za GET funkciju koja obezbedjuje pribavljanje lokalne promenljive sa poslednjom vrednošću.

Dockerfile:

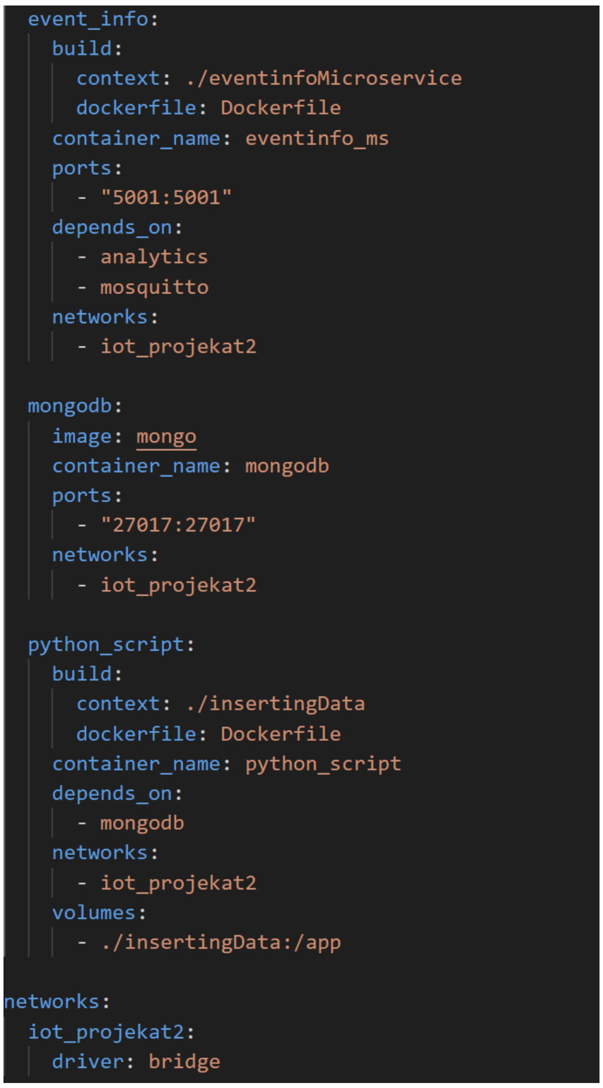


```dockerfile
FROM node:14

WORKDIR /app

COPY package.json package-lock.json ./
RUN npm install

COPY . .

EXPOSE 5001

CMD ["node", "app.js"]
```

Docker-compose file

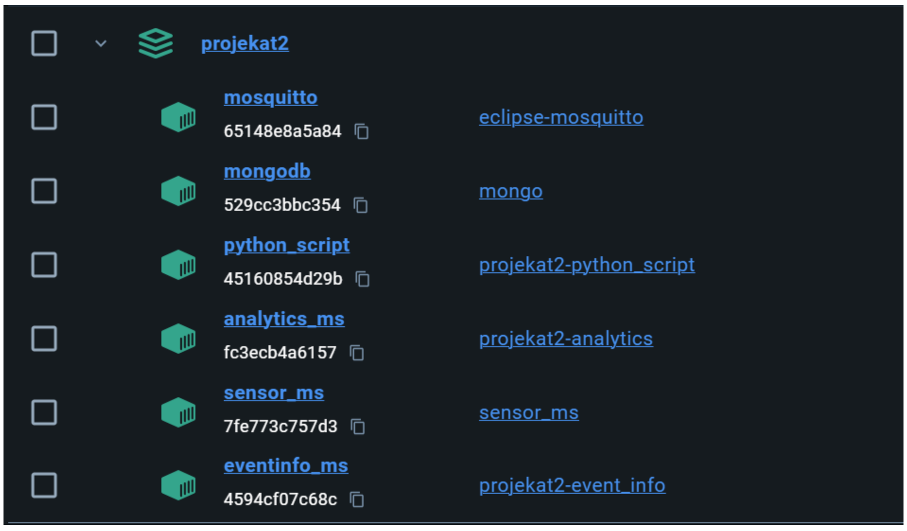


```yaml
version: "3.7"

services:
  mosquitto:
    image: eclipse-mosquitto
    hostname: mosquitto
    container_name: mosquitto
    restart: unless-stopped
    ports:
      - "1883:1883"
      - "9001:9001"
    volumes:
      - ./mosquitto.conf:/mosquitto/config/mosquitto.conf
    networks:
      - iot_projekat2

  sensor:
    image: sensor_ms
    container_name: sensor_ms
    ports:
      - "3000:3000"
    networks:
      - iot_projekat2
    depends_on:
      - mosquitto
      - mongodb
      - python_script

  analytics:
    build:
      context: ./analyticsmicroservice
      dockerfile: Dockerfile
    container_name: analytics_ms
    depends_on:
      - mongodb
      - python_script
    networks:
      - iot_projekat2
```

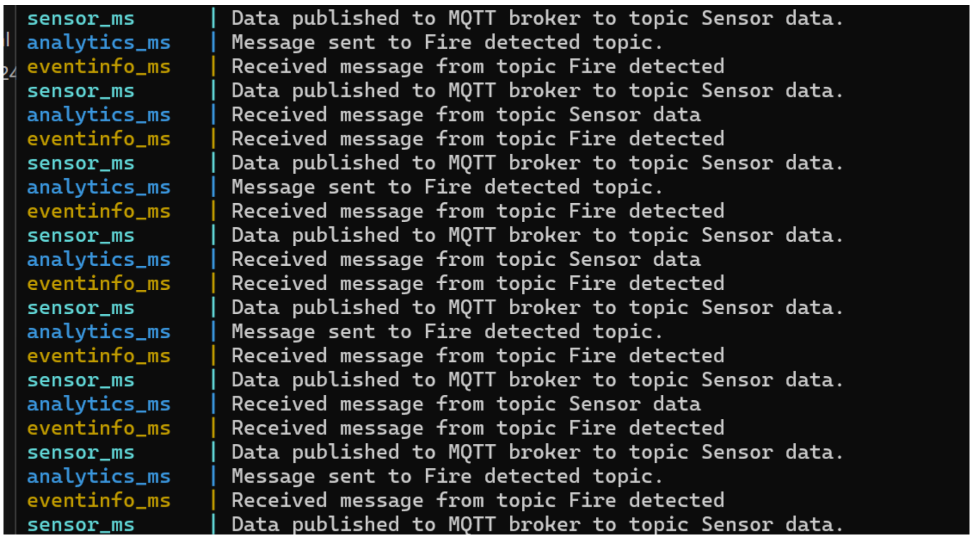


```yaml
  event_info:
    build:
      context: ./eventinfoMicroservice
      dockerfile: Dockerfile
    container_name: eventinfo_ms
    ports:
      - "5001:5001"
    depends_on:
      - analytics
      - mosquitto
    networks:
      - iot_projekat2

  mongodb:
    image: mongo
    container_name: mongodb
    ports:
      - "27017:27017"
    networks:
      - iot_projekat2

  python_script:
    build:
      context: ./insertingData
      dockerfile: Dockerfile
    container_name: python_script
    depends_on:
      - mongodb
    networks:
      - iot_projekat2
    volumes:
      - ./insertingData:/app

networks:
  iot_projekat2:
  driver: bridge
```

Naredbom *docker-compose up* se kreiraju sledeći container-i:



```
projekat2
  mosquitto        65148e8a5a84    eclipse-mosquitto
  mongodb          529cc3bbc354    mongo
  python_script    45160854d29b    projekat2-python_script
  analytics_ms     fc3ecb4a6157    projekat2-analytics
  sensor_ms        7fe773c757d3    sensor_ms
  eventinfo_ms     4594cf07c68c    projekat2-event_info
```

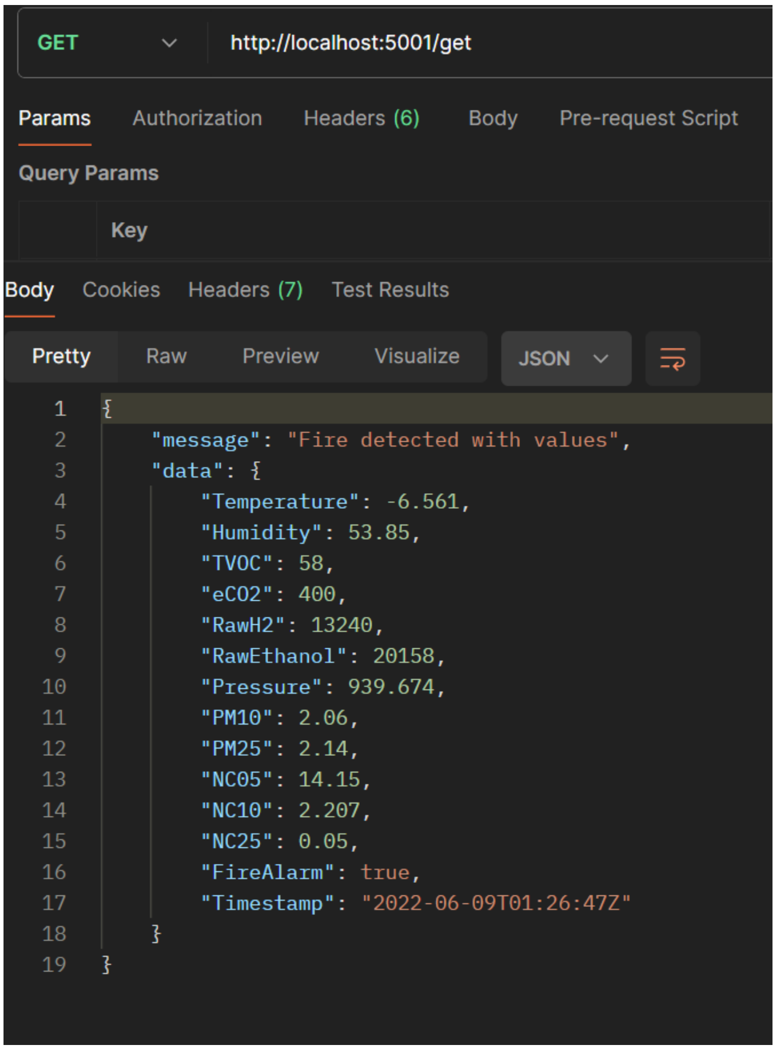


```
sensor_ms         | Data published to MQTT broker to topic Sensor data.
analytics_ms      | Message sent to Fire detected topic.
eventinfo_ms      | Received message from topic Fire detected
sensor_ms         | Data published to MQTT broker to topic Sensor data.
analytics_ms      | Received message from topic Sensor data
eventinfo_ms      | Received message from topic Fire detected
sensor_ms         | Data published to MQTT broker to topic Sensor data.
analytics_ms      | Message sent to Fire detected topic.
eventinfo_ms      | Received message from topic Fire detected
sensor_ms         | Data published to MQTT broker to topic Sensor data.
analytics_ms      | Received message from topic Sensor data
eventinfo_ms      | Received message from topic Fire detected
sensor_ms         | Data published to MQTT broker to topic Sensor data.
analytics_ms      | Message sent to Fire detected topic.
eventinfo_ms      | Received message from topic Fire detected
sensor_ms         | Data published to MQTT broker to topic Sensor data.
analytics_ms      | Received message from topic Sensor data
eventinfo_ms      | Received message from topic Fire detected
sensor_ms         | Data published to MQTT broker to topic Sensor data.
analytics_ms      | Message sent to Fire detected topic.
eventinfo_ms      | Received message from topic Fire detected
sensor_ms         | Data published to MQTT broker to topic Sensor data.
```

Nakon pokretanja u konzoli se mogu videti logovi vezani za komunikaciju preko MQTT brokera.

Testiranje putem Postman-a

Testiranje se vrši na adresi koju obezbedjuje EventInfo mikroservis.



GET http://localhost:5001/get

```json
{
    "message": "Fire detected with values",
    "data": {
        "Temperature": -6.561,
        "Humidity": 53.85,
        "TVOC": 58,
        "eCO2": 400,
        "RawH2": 13240,
        "RawEthanol": 20158,
        "Pressure": 939.674,
        "PM10": 2.06,
        "PM25": 2.14,
        "NC05": 14.15,
        "NC10": 2.207,
        "NC25": 0.05,
        "FireAlarm": true,
        "Timestamp": "2022-06-09T01:26:47Z"
    }
}
```