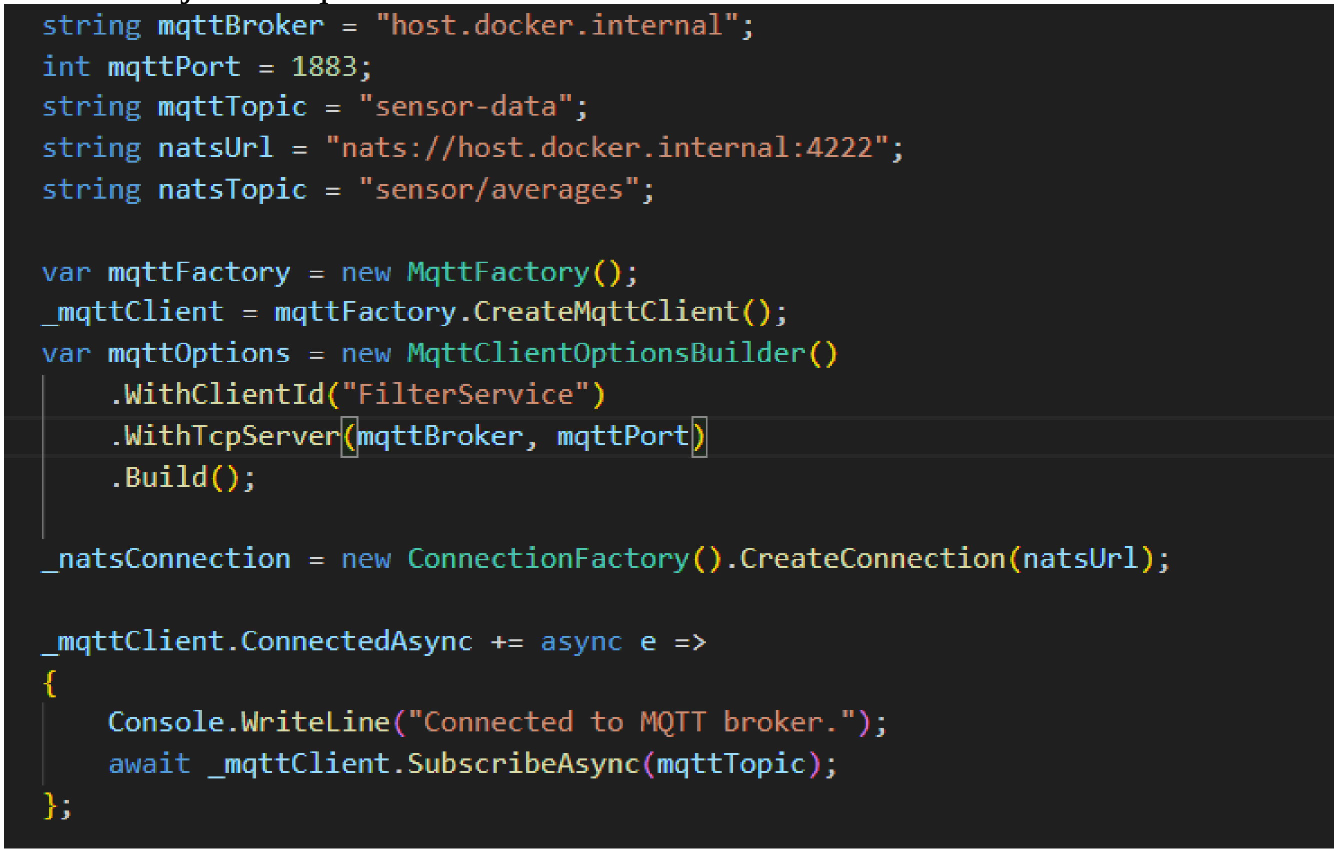
Internet stvari i servisa 3. projekat

Tri mikroservisa implementirana u različitim tehnologijama.
Mikroservisi

- Filter
- Dashboard
- Command

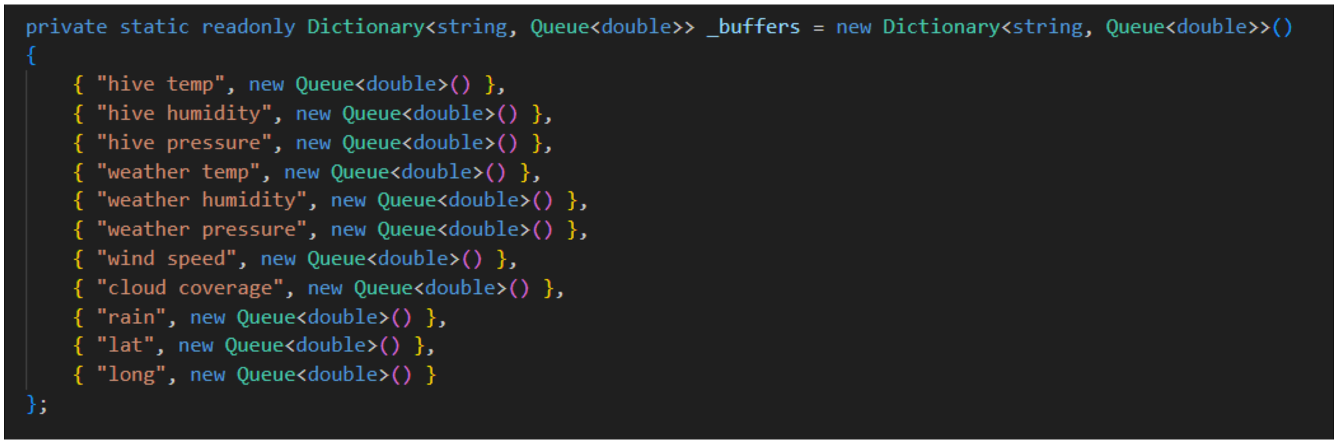
Filter

Rađen u .NET-u
Povezivanje na mqtt I nats



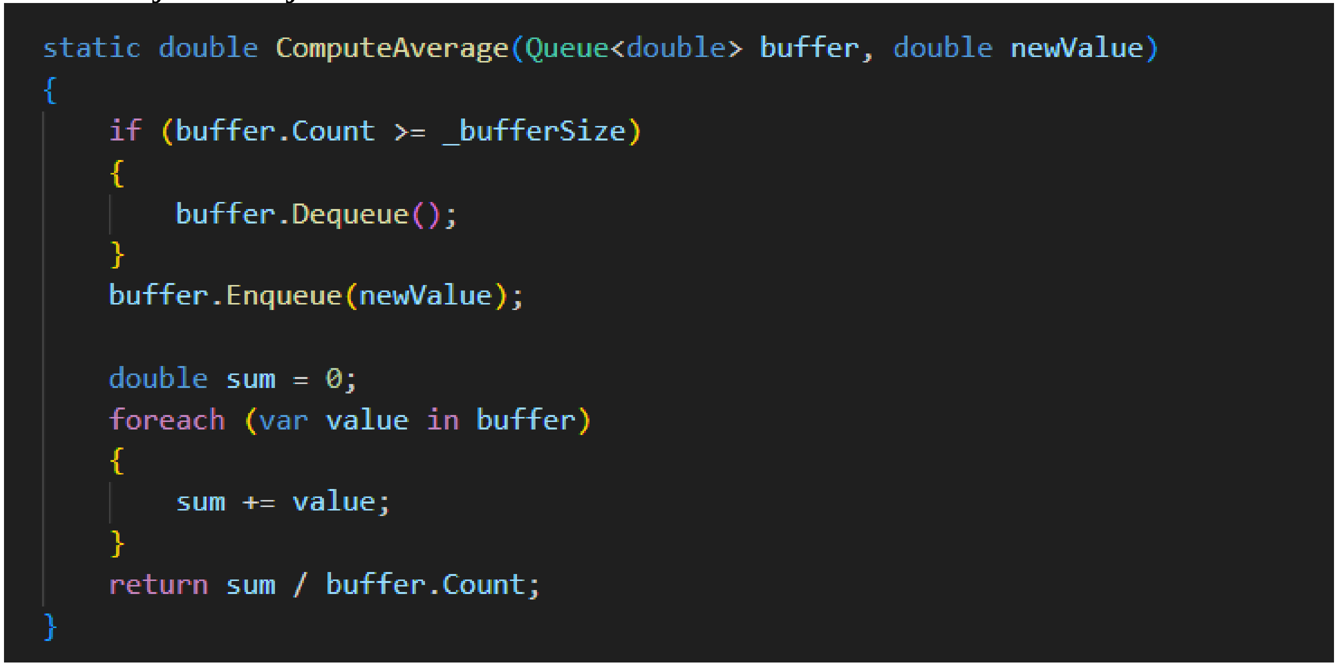
```
string mqttBroker = "host.docker.internal";
int mqttPort = 1883;
string mqttTopic = "sensor-data";
string natsUrl = "nats://host.docker.internal:4222";
string natsTopic = "sensor/averages";

var mqttFactory = new MqttFactory();
_mqttClient = mqttFactory.CreateMqttClient();
var mqttOptions = new MqttClientOptionsBuilder()
    .WithClientId("FilterService")
    .WithTcpServer(mqttBroker, mqttPort)
    .Build();

_natsConnection = new ConnectionFactory().CreateConnection(natsUrl);

_mqttClient.ConnectedAsync += async e =>
{
    Console.WriteLine("Connected to MQTT broker.");
    await _mqttClient.SubscribeAsync(mqttTopic);
};
```

Vrednosti atributa



```
private static readonly Dictionary<string, Queue<double>> _buffers = new Dictionary<string, Queue<double>>()
{
    { "hive temp", new Queue<double>() },
    { "hive humidity", new Queue<double>() },
    { "hive pressure", new Queue<double>() },
    { "weather temp", new Queue<double>() },
    { "weather humidity", new Queue<double>() },
    { "weather pressure", new Queue<double>() },
    { "wind speed", new Queue<double>() },
    { "cloud coverage", new Queue<double>() },
    { "rain", new Queue<double>() },
    { "lat", new Queue<double>() },
    { "long", new Queue<double>() }
};
```

Računanje srednje vrednosti

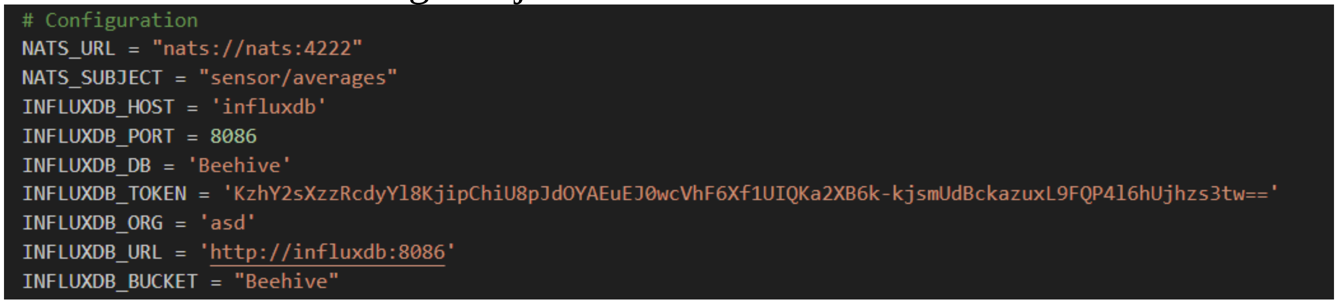


```
static double ComputeAverage(Queue<double> buffer, double newValue)
{
    if (buffer.Count >= _bufferSize)
    {
        buffer.Dequeue();
    }
    buffer.Enqueue(newValue);

    double sum = 0;
    foreach (var value in buffer)
    {
        sum += value;
    }
    return sum / buffer.Count;
}
```

Dashboard

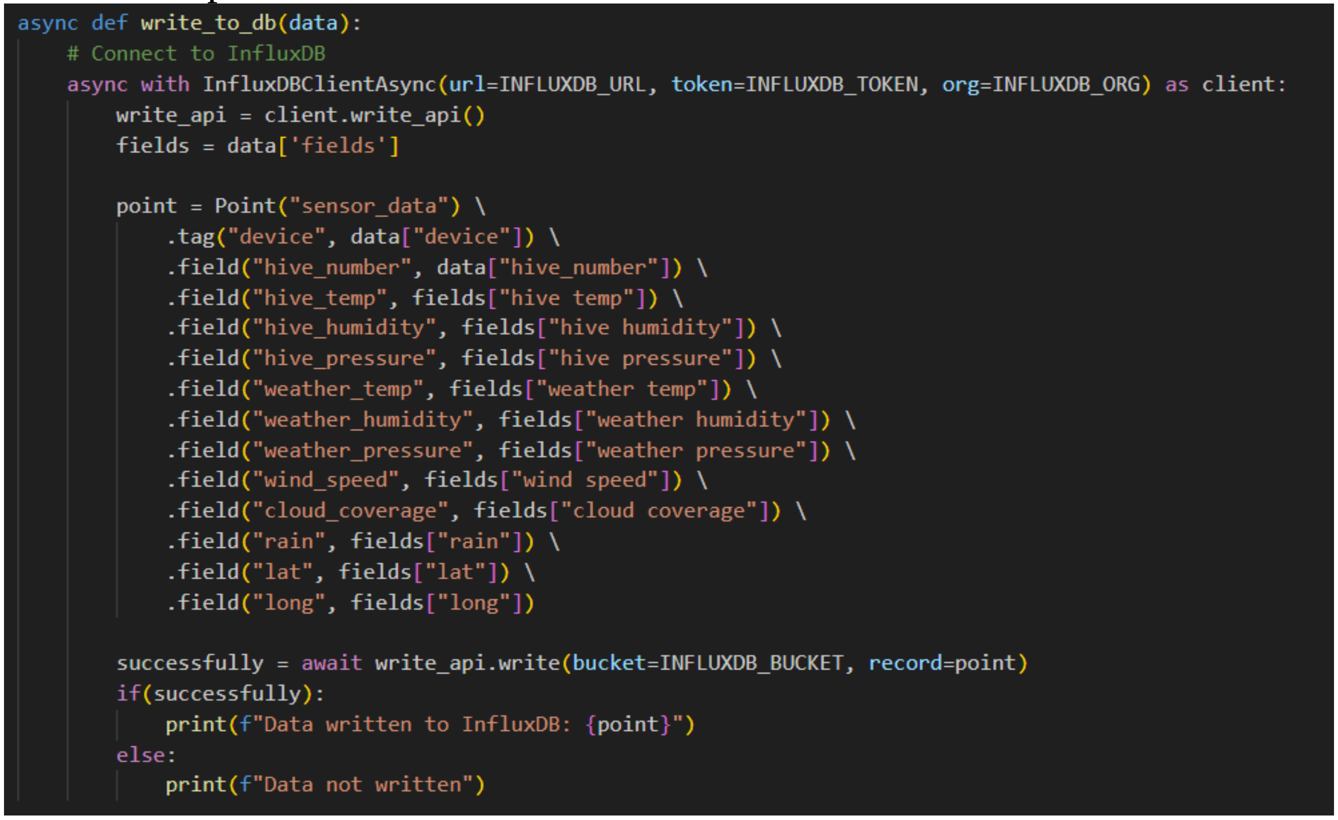
Rađen u pythonu

NATS I influxDB konfiguracija



```python
# Configuration
NATS_URL = "nats://nats:4222"
NATS_SUBJECT = "sensor/averages"
INFLUXDB_HOST = 'influxdb'
INFLUXDB_PORT = 8086
INFLUXDB_DB = 'Beehive'
INFLUXDB_TOKEN = 'KzhY2sXzzRcdyYl8KjipChiU8pJdOYAEuEJ0wcVhF6Xf1UIQKa2XB6k-kjsmUdBckazuxL9FQP4l6hUjhzs3tw=='
INFLUXDB_ORG = 'asd'
INFLUXDB_URL = 'http://influxdb:8086'
INFLUXDB_BUCKET = "Beehive"
```

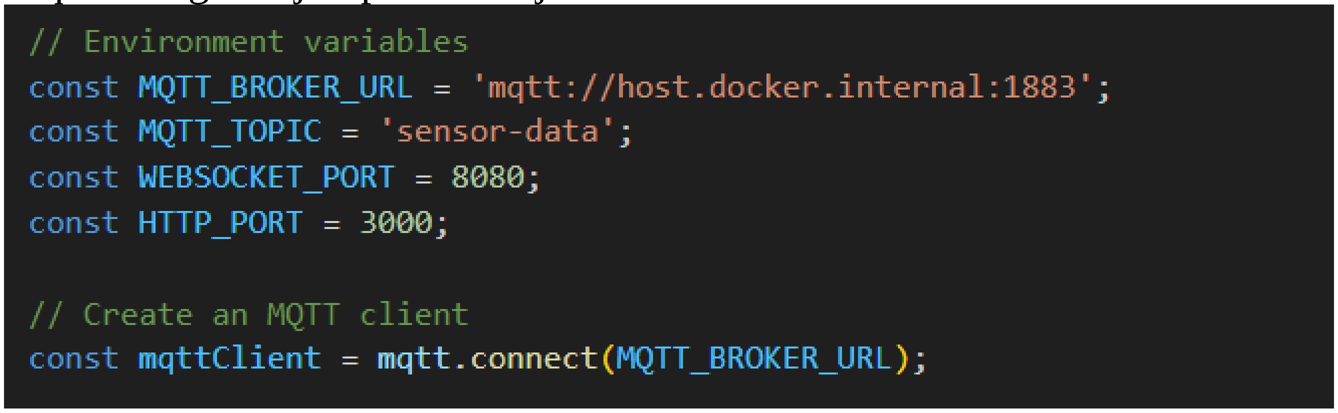
Asinhroni upis u bazu



```python
async def write_to_db(data):
    # Connect to InfluxDB
    async with InfluxDBClientAsync(url=INFLUXDB_URL, token=INFLUXDB_TOKEN, org=INFLUXDB_ORG) as client:
        write_api = client.write_api()
        fields = data['fields']

        point = Point("sensor_data") \
            .tag("device", data["device"]) \
            .field("hive_number", data["hive_number"]) \
            .field("hive_temp", fields["hive temp"]) \
            .field("hive_humidity", fields["hive humidity"]) \
            .field("hive_pressure", fields["hive pressure"]) \
            .field("weather_temp", fields["weather temp"]) \
            .field("weather_humidity", fields["weather humidity"]) \
            .field("weather_pressure", fields["weather pressure"]) \
            .field("wind_speed", fields["wind speed"]) \
            .field("cloud_coverage", fields["cloud coverage"]) \
            .field("rain", fields["rain"]) \
            .field("lat", fields["lat"]) \
            .field("long", fields["long"])

        successfully = await write_api.write(bucket=INFLUXDB_BUCKET, record=point)
        if(successfully):
            print(f"Data written to InfluxDB: {point}")
        else:
            print(f"Data not written")
```

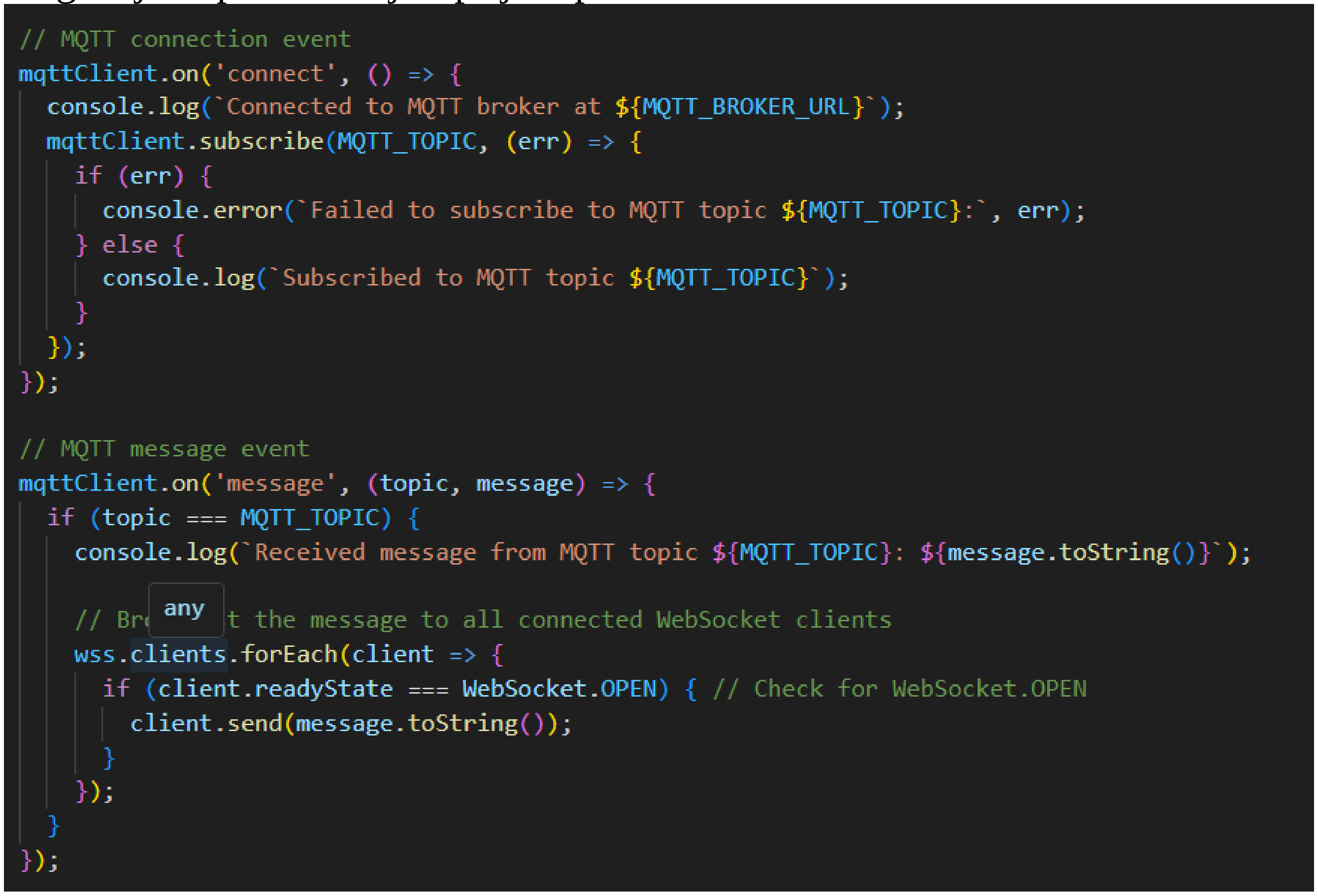
Command

Rađen u nodejs
Mqtt konfiguracija I povezivanje



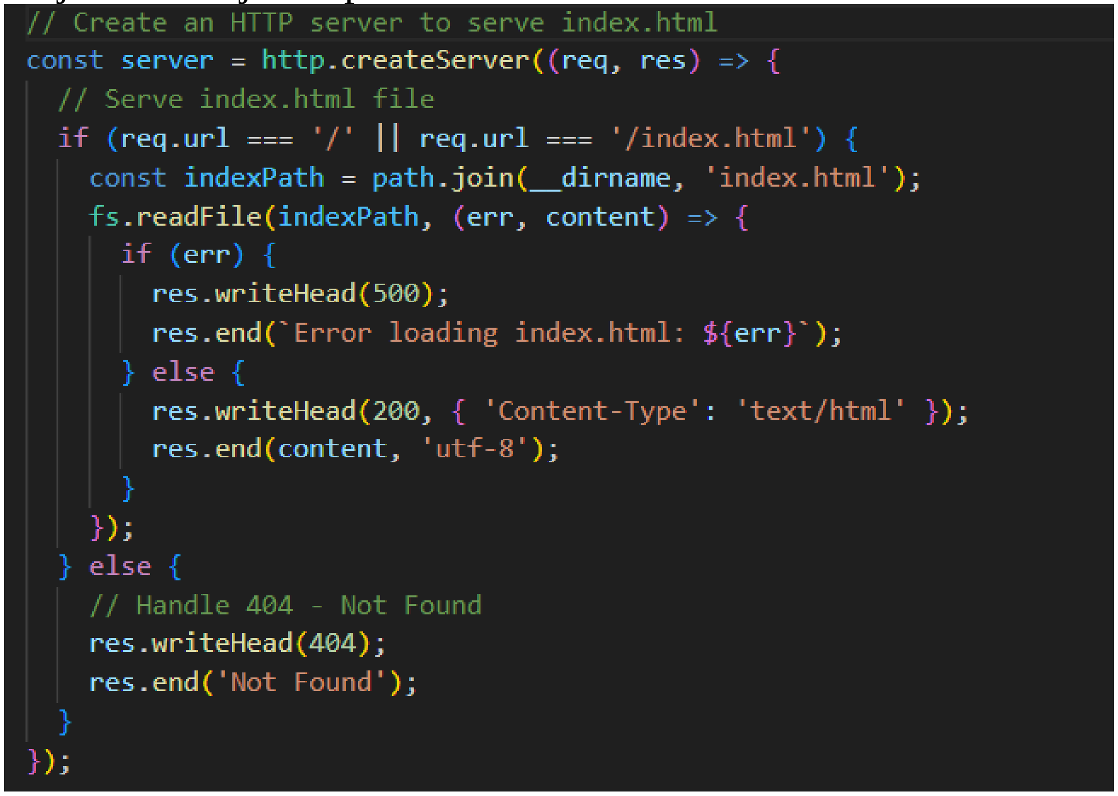
```
// Environment variables
const MQTT_BROKER_URL = 'mqtt://host.docker.internal:1883';
const MQTT_TOPIC = 'sensor-data';
const WEBSOCKET_PORT = 8080;
const HTTP_PORT = 3000;

// Create an MQTT client
const mqttClient = mqtt.connect(MQTT_BROKER_URL);
```

Događaji za povezivanje I prijem poruke



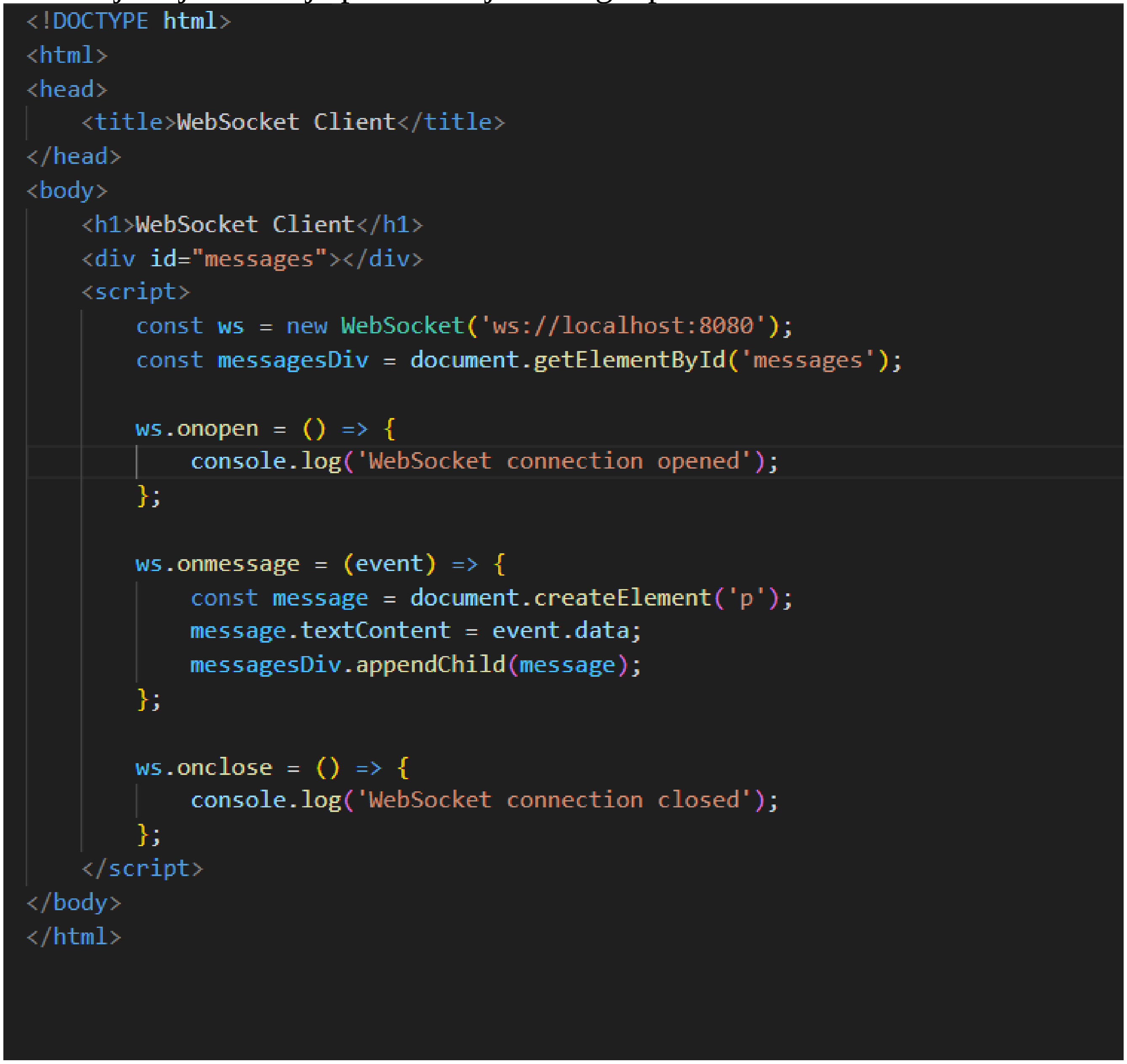
```
// MQTT connection event
mqttClient.on('connect', () => {
  console.log(`Connected to MQTT broker at ${MQTT_BROKER_URL}`);
  mqttClient.subscribe(MQTT_TOPIC, (err) => {
    if (err) {
      console.error(`Failed to subscribe to MQTT topic ${MQTT_TOPIC}:`, err);
    } else {
      console.log(`Subscribed to MQTT topic ${MQTT_TOPIC}`);
    }
  });
});

// MQTT message event
mqttClient.on('message', (topic, message) => {
  if (topic === MQTT_TOPIC) {
    console.log(`Received message from MQTT topic ${MQTT_TOPIC}: ${message.toString()}`);

    // Br any t the message to all connected WebSocket clients
    wss.clients.forEach(client => {
      if (client.readyState === WebSocket.OPEN) { // Check for WebSocket.OPEN
        client.send(message.toString());
      }
    });
  }
});
```

“Serviranje” html fajla na port 3000



```javascript
// Create an HTTP server to serve index.html
const server = http.createServer((req, res) => {
  // Serve index.html file
  if (req.url === '/' || req.url === '/index.html') {
    const indexPath = path.join(__dirname, 'index.html');
    fs.readFile(indexPath, (err, content) => {
      if (err) {
        res.writeHead(500);
        res.end(`Error loading index.html: ${err}`);
      } else {
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.end(content, 'utf-8');
      }
    });
  } else {
    // Handle 404 - Not Found
    res.writeHead(404);
    res.end('Not Found');
  }
});
```

Html fajl koji hendluje poruke koje su stigle preko web soketa



```html
<!DOCTYPE html>
<html>
<head>
    <title>WebSocket Client</title>
</head>
<body>
    <h1>WebSocket Client</h1>
    <div id="messages"></div>
    <script>
        const ws = new WebSocket('ws://localhost:8080');
        const messagesDiv = document.getElementById('messages');

        ws.onopen = () => {
            console.log('WebSocket connection opened');
        };

        ws.onmessage = (event) => {
            const message = document.createElement('p');
            message.textContent = event.data;
            messagesDiv.appendChild(message);
        };

        ws.onclose = () => {
            console.log('WebSocket connection closed');
        };
    </script>
</body>
</html>
```