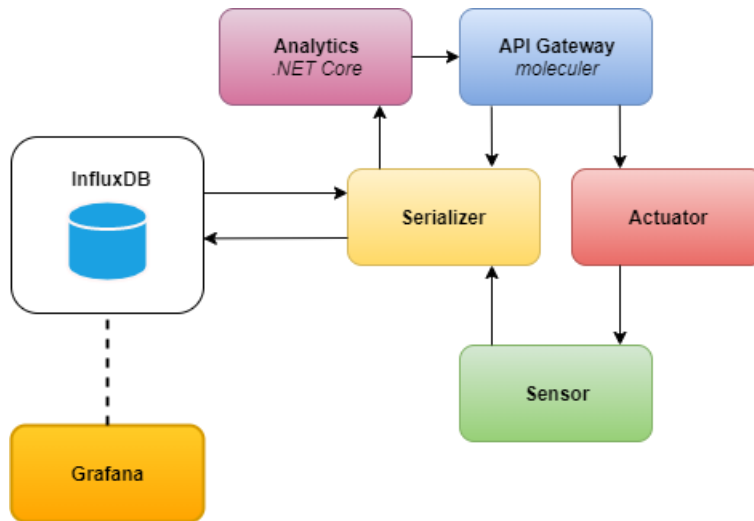


Mikroservisi i IoT - nastavak

Primer sa prethodnog časa je proširen tako da sada sadrži i Analytics servis napisan u .NET Core-u. Arhitektura primera je prikazana na sledećoj slici:



1. API – nudi rute za čitanje temperature sa odgovarajućeg senzora i za postavljanje offset-a senzoru sa odgovarajućim id-jem.
2. Serializer – mikroservis koji ostvaruje konekciju ka bazi podataka i vrši upis i čitanje podataka o temperaturi.
3. InfluxDB – je baza otvorenog koda namenjena za rad sa vremenskim serijama.
4. Senzor – dummy senzor koji simulira podatke o temperaturi i šalje nova „merenja“ svake sekunde. Takođe, sluša na promene u offset-u od aktuatora.
5. Aktuator – šalje odgovarajućem senzoru (sa zadatim id-jem) novi offset.
6. Grafana – alat za vizuelizaciju podataka iz InfluxDB-a.
7. Analytics – prikuplja podatke sa senzora i ukoliko detektuje preveliku temperaturu, šalje zahtev za ažuriranje offset-a preko gateway-a.

Da bi aplikacija mogla da se doradi tako da šalje zahtev Analytics servisu, moleculer aplikacija treba da ima instaliran request modul što se postiže izvršavanjem komande **npm install request**.

Delovi koji su nepromenjeni neće biti prikazani u ovom dokumentu.

actuator.service.js - izmenjeno je ime parametra koji se prosleđuje, da bi bilo u skladu sa body-jem koji stiže iz Analytics dela (sensorId)

```
actuator.service.js X
moleculer-iot > services > actuator.service.js > ...
1  "use strict";
2
3  module.exports = {
4    name: "actuator",
5    actions: {
6      set: {
7        params: {
8          offset: { type: "number" },
9          sensorId: { type: "number" }
10       },
11       async handler(ctx) {
12         console.log(ctx.params);
13         this.broker.emit(`temperature.set.${ctx.params.sensorId}`, ctx.params);
14         return 'Success setting temperature!';
15       }
16     }
17   }
18 };
```

serializer.service.js – dodata je nova metoda sendTemperature, koja se poziva prilikom detektovanja temperature.read eventa, odnosno, kada je pročitana nova temperatura sa senzora, pored toga što se upiše u bazu, poslaće se i analytics servisu, korišćenjem post zahteva iz request biblioteke. Šalje se izmerena temperatura, kao i id senzora na kom je izmerena. Kada se dobije odgovor, ukoliko je došlo do greške, loguje se ta greška, ukoliko nije, štampa se status kod i body. Omogućava povezivanje sa influx bazom podataka, nudi akciju za čitanje svih merenja iz baze temperatura za odgovarajući senzor, a ima i event koji upisuje merenja sa senzora u bazu kada je emitovan temperature.read event.

```

methods: {
  sendTemperature(temp, sensorId) {
    const body = {
      temperature: temp,
      sensorId: sensorId,
      offset: 0
    };
    console.log(body);
    request.post(process.env.ANALYTICS_URL, {
      json: body
    }, (err, res, body) => {
      if (err) {
        console.log(err);
        return;
      }
      console.log(res.statusCode);
      console.log(body);
    });
  }
},

```

```

this.influx.writePoints([
  {
    measurement: 'temperature',
    fields: {
      temperature: payload.temperature,
      sensorId: payload.sensorId
    },
    time: payload.timestamp
  }
]);
this.sendTemperature(payload.temperature, payload.sensorId);

```

Created metoda je ostala nepromenjena u serialize.service.js fajlu.

gateway.service.js – rute GET /temperature i PUT /set

PUT metoda iz gateway servisa se poziva onog trenutka kada je analytics servis zaključio da treba da koriguje temperaturu, te će na ovu rutu da pošalje novi offset i sensorId kome se šalje. Ovi podaci se kao i ranije prosleđuju aktuatoru.

Analytics

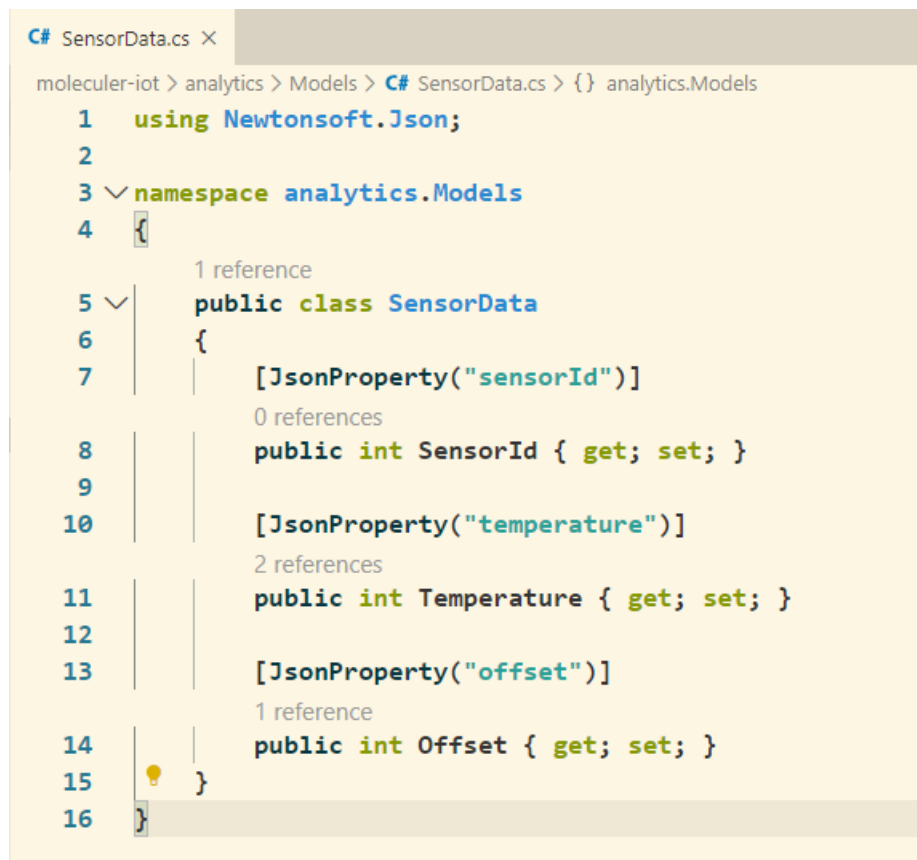
Da bismo kreirali mikroservis u .NET Core-u, iskoristićemo kao i na prethodnim časovima .NET sdk. Komanda koja se poziva za kreiranje je:

- ***dotnet new webapi --o analytics --no-https***

Od paketa je neophodno instalirati Newtonsoft.Json, koji će biti korišćen za serijalizaciju i ovo se postiže komandom:

- ***dotnet add package Newtonsoft.Json***

U kreiranom dotnet projektu postoji dosta već gotovog koda koji za nas nije od značaja. Da bismo kreirali funkcionalnosti koje treba da podrži ovaj servis potreban nam je prvo model SensorData koji će biti smešten u Models direktorijumu. Ovaj model će sadržati sve što treba da se prosledi analytics kontroleru i izgleda ovako:



```
C# SensorData.cs X
moleculer-iot > analytics > Models > C# SensorData.cs > {} analytics.Models
1  using Newtonsoft.Json;
2
3  namespace analytics.Models
4  {
5      1 reference
6      public class SensorData
7      {
8          [JsonProperty("sensorId")]
9          0 references
10         public int SensorId { get; set; }
11
12         [JsonProperty("temperature")]
13         2 references
14         public int Temperature { get; set; }
15
16         [JsonProperty("offset")]
17         1 reference
18         public int Offset { get; set; }
19     }
20 }
```

Postavljanjem JsonProperty-ja na offset napominjemo da će se u okviru JSON objekta offset navoditi malim početnim slovom.

Dodajemo Analysis.cs fajl u Controllers direktorijumu.



```
10
11 namespace analytics.Controllers
12 {
13     [ApiController]
14     [Route("/api/[controller]")]
15     public class AnalysisController : ControllerBase
16     {
17         private static readonly int TMax = 35;
18         public AnalysisController()
19         {
20         }
21
22         [HttpGet]
23         public ActionResult<string> Get() {
24             return Ok("Analysis works!");
25         }
```

Get metoda je kreirana samo testiranja radi i kao što je prikazano na slici vraća poruku da Analysis radi.

Post metoda na `/api/analysis` će prvo izračunati da li treba izvršiti korekciju offseta i ako treba za koliko treba smanjiti temperaturu. Kao telo ove metode nalazi se objekat tipa `SensorData` koji u sebi sadrži temperaturu, id senzora i offset. Ukoliko je izmerena temperatura veća od maksimalne definisane, računa se novi offset koji se prosleđuje API Gateway-u. Ukoliko je offset različit od nule treba poslati zahtev, u suprotnom je dovoljno vratiti 200 OK odgovor sa porukom da je temperatura u redu. Za slučaj da je offset različit od nule, ovom objektu koji je dobijen se ažurira offset i taj objekat se serijalizuje i prosleđuje pozivanjem PUT zahteva na <http://gateway:3000/set>. Navodi se gateway zato što je to ime kontejnera koje smo naveli ranije u `docker-compose.yml` fajlu. Kada je stigao odgovor, testiranja radi ova metoda pored poruke da je temperatura korigovana, vraća i odgovor koji je dobila sa gateway-a.

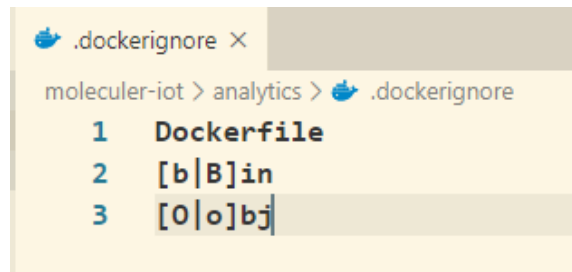
```

[HttpPost]
0 references
public async Task<IActionResult> Post(SensorData data)
{
    int offset = data.Temperature > TMax ? TMax - data.Temperature : 0;
    if (offset != 0) {
        using (var httpClient = new HttpClient())
        {
            data.Offset = offset;
            var c = JsonConvert.SerializeObject(data);
            StringContent content = new StringContent(c, Encoding.UTF8, "application/json");
            using (var response = await httpClient.PutAsync("http://gateway:3000/set", content))
            {
                string apiResponse = await response.Content.ReadAsStringAsync();
                return new JsonResult(
                    new
                    {
                        resp = apiResponse,
                        message = $"Temperature corrected - {offset}!",
                    }
                );
            }
        }
    }
    return new JsonResult(
        new
        {
            message = "Temperature OK!"
        }
    );
}

```

Potrebno je kreirati Dockerfile i .dockerignore fajl za analytics projekat.

.dockerignore



```

moleculer-iot > analytics > .dockerignore
1  Dockerfile
2  [b|B]in
3  [O|o]bj

```

Dockerfile

Dockerfile će biti kreiran na osnovu .NET Core slike, kao što je prikazano na prethodnim časovima. Prikazan je na sledećoj slici.

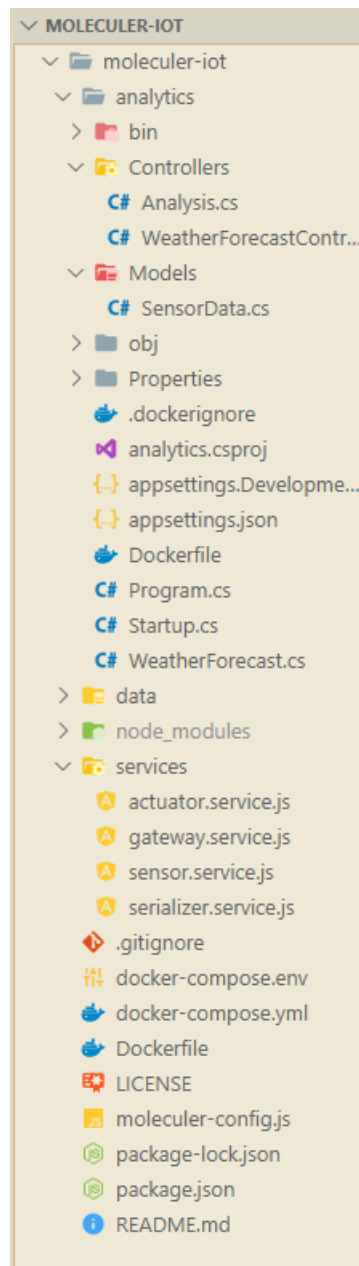
```
Dockerfile X
moleculer-iot > analytics > Dockerfile > ...
1 FROM mcr.microsoft.com/dotnet/core/sdk:3.1 AS build
2 WORKDIR /src
3 COPY analytics.csproj .
4 COPY . .
5 RUN dotnet restore
6 RUN dotnet publish -c release -o /app
7
8 FROM mcr.microsoft.com/dotnet/core/aspnet:3.1
9 WORKDIR /app
10 COPY --from=build /app .
11 ENTRYPOINT ["dotnet", "analytics.dll"]
```

Treba proširiti docker-compose.env fajl sa prošlog časa, tako da sadrži i URL za ANALYTICS servis koji je eksteran u odnosu na aplikaciju u Moleculer-u.

```
docker-compose.env X
moleculer-iot > docker-compose.env
1 LOGLEVEL=info
2 SERVICEDIR=services
3 TRANSPORTER=nats://nats:4222
4 ANALYTICS_URL=http://analytics/api/Analysis
```

Dodaje se konstanta ANALYTICS_URL koja umesto localhost sadrži analytics kao ime kontejnera koje će biti zadato ovom servisu.

Struktura projekta sada ovako izgleda:



`docker-compose.yml`

Ovaj fajl se proširava tako što se dodaje analytics kontejner koji će se kreirati na osnovu slike opisane Dockefile-om koji se ne nalazi u istom direktorijumu kao docker-compose.yml fajl, pa se zato ne navodi tačka u build delu već ./analytics. Navodi se i da će servis biti dostupan na portu 5000 (port 3000 je zauzet API Gateway-em).


```
analytics:
  build:
    context: ./analytics
  image: analytics
  ports:
    - "5000:80"
```

Pokrećemo naše kontejnere korišćenjem naredbe **docker-compose up --build**.

Početak izvršenja naredbe:

```
D:\moleculer-iot>docker-compose up --build
Building gateway
Step 1/7 : FROM node:latest
--> c31fbeb964cc
Step 2/7 : RUN mkdir /app
--> Using cache
--> 54072c727ea0
Step 3/7 : WORKDIR /app
--> Using cache
--> c2188506b331
Step 4/7 : ADD package*.json /app/
--> Using cache
--> 7a3a71d3029a
Step 5/7 : RUN npm install
--> Using cache
--> 4dc7e15f40ba
Step 6/7 : ADD . /app/
--> 0b4098fb5f36
Step 7/7 : CMD [ "npm", "start" ]
--> Running in 049659e1850f
Removing intermediate container 049659e1850f
--> 0236f3df7f77
Successfully built 0236f3df7f77
Successfully tagged service-gateway:latest
Building serializer
```

Testiranje

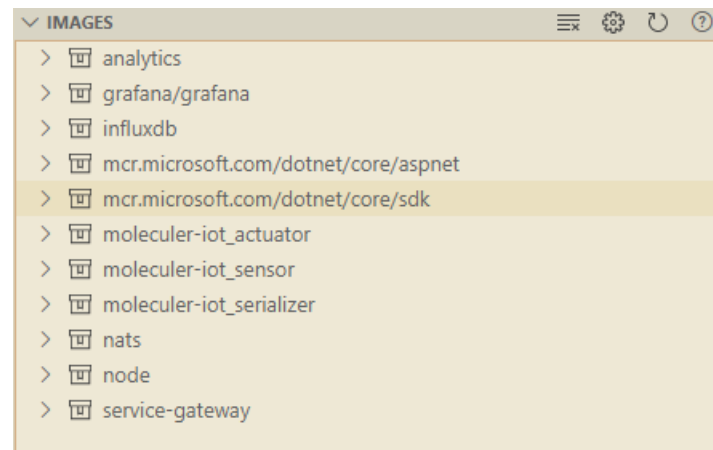
1. Senzor 1 emituje temperature.read sa temperaturom 39.
2. Serializer koji sluša promene temperature upisuje u bazu ovu temperaturu i šalje POST zahtev na /api/analysis sa izmerenom temperaturom.
3. Analytics konstatuje da je temperatura previsoka i šalje zahtev da se promeni offset, koji stiže prvo do gateway-a kome je i upućen.
4. Zatim gateway prosleđuje novi offset aktuatoru.
5. Aktuator onda prosleđuje novi offset senzoru sa zadatim id-jem.
6. Do serializer-a konačno stiže odgovor sa statusom 200 i vidi se koji je novi offset, kao i da je resp bio *Success setting temperature!*, što je postavljeno kao odgovor u aktuatoru.

```

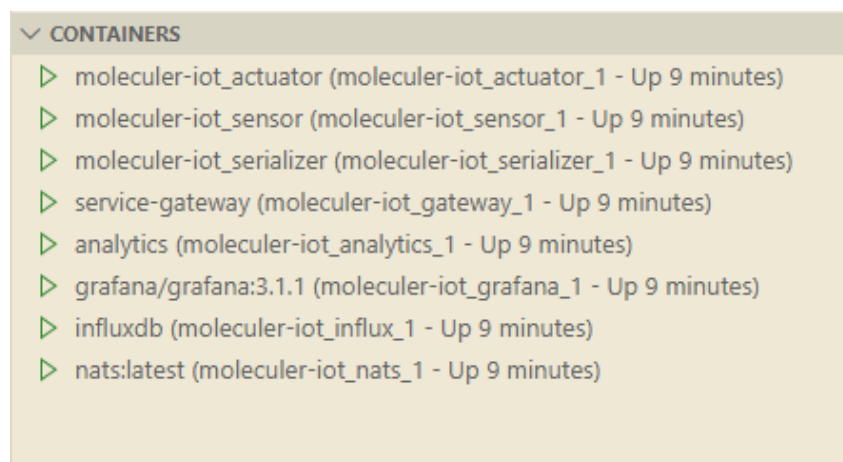
serializer_1 | Recieved "temperature.read" event in serializer service with payload: { sensorId: 1, temperature: 39, timestamp: 158942336371 }
serializer_1 | { temperature: 39, sensorId: 1, offset: 0 }
influx_1      | [httpd] 172.22.0.9 - admin [14/May/2020:02:28:56 +0000] "POST /write?db=temperature&p=%5BRREDACTED%5D&precision=n&rp=&u=admin HTTP/1.1" 204 0 "-" "-"
a9211dba-958a-11ea-8005-0242ac160003 27314
gateway_1    | { sensorId: 1, temperature: 39, offset: -4 }
actuator_1   | { sensorId: 1, temperature: 39, offset: -4 }
sensor_1     | Recieved "temperature.set.1" event in sensor service with payload: { sensorId: 1, temperature: 39, offset: -4 }
serializer_1 | 200
serializer_1 | {
serializer_1 |   resp: 'Success setting temperature!',
serializer_1 |   message: 'Temperature corrected - -4!'
serializer_1 | }

```

docker slike



docker kontejneri



GET na /api/analysis

GET http://localhost:5000/api/analysis + ... No Environment

Untitled Request Co

GET http://localhost:5000/api/analysis Send

Params Authorization Headers (10) Body ● Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

body Cookies (1) Headers (4) Test Results Status: 200 OK Time: 36 ms Size: 157 B Save

Pretty Raw Preview Visualize Text ↺

```
1 Analysis works!
```

Github repozitorijum u kome se nalazi kod je moguće videti [OVDE](#).