

SERVISNO-ORIENTISANE ARHITEKTURE PROJEKAT

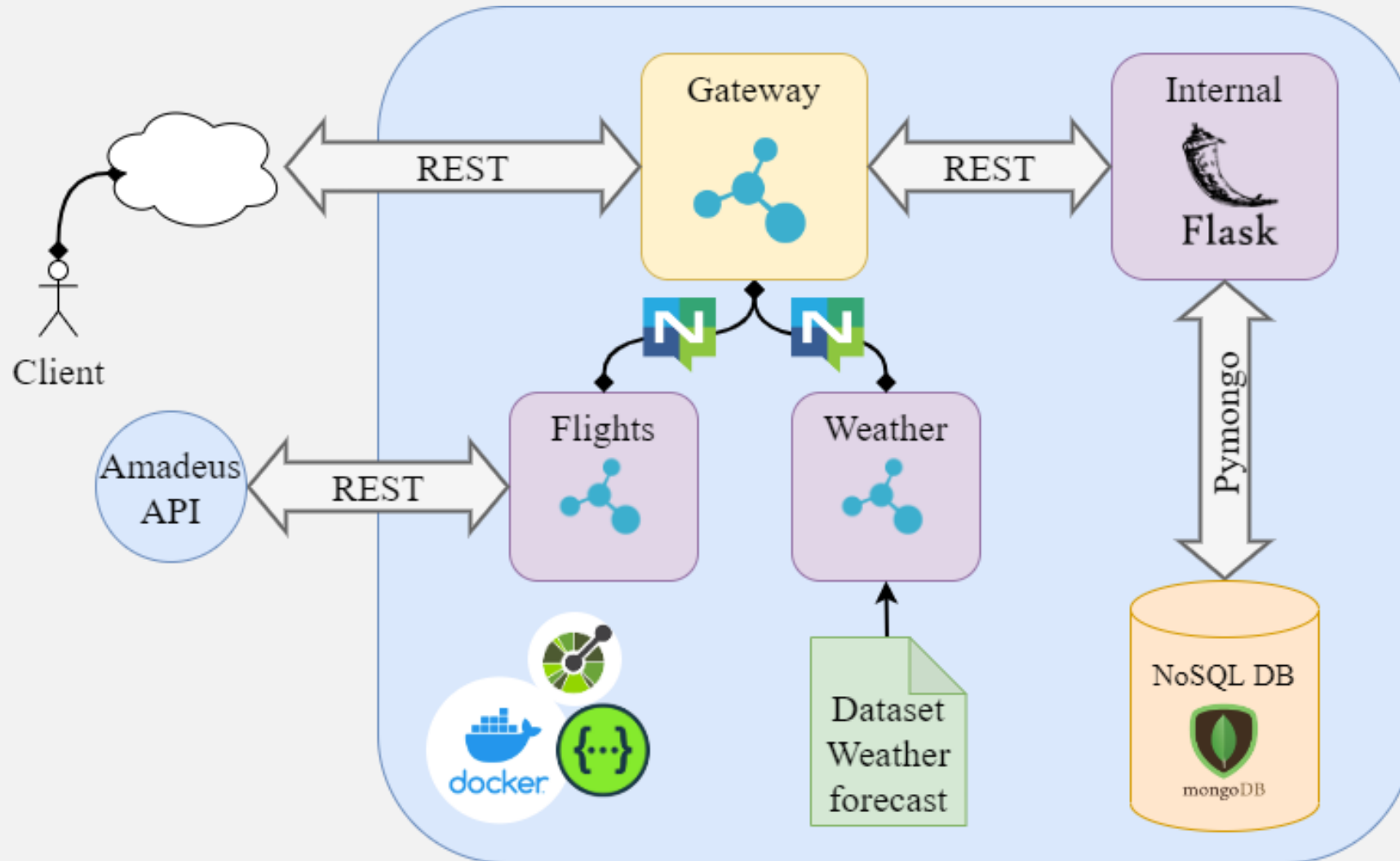
Prof. dr Dragan Stojanović

Tim Swiftly
Teodora Kocić 17410
Stefan Aleksić 16995

PROJEKAT I

- Amadeus API je odabran za public API projekta.
 - Pruža razmenu informacija sa velikim turističkim organizacijama.
 - Omogućava pretragu destinacija, aerodromova, letova, hotela, kao i samu rezervaciju karata.
- Za dataset su odabrani podaci o vremenskoj prognozi, koji simuliraju očitavanja prognoze u realnom vremenu.

PROJEKAT I - ARHITEKTURA




PROJEKAT I - OPIS

- Svi servisi su pokrenuti uz pomoć docker-compose, u okviru sopstvenih kontejnera.
- Gateway, Flight i Weather service-i su pisani kao Moleculer service-i u okviru Nodejs skriptnih fajlova, a za broker između servisa je podrazumevano postavljen NATS.
- Internal API je implementiran u Python-u uz pomoć biblioteka Flask i pymongo, koje olakšavaju REST komunikaciju, odnosno komunikaciju sa bazom, respektivno.
- Za NoSQL bazu je odabrana document store-a baza podatak MongoDB.

PROJEKAT I – TOK PODATAKA

- Flights service, uz pomoć Amadeus javno dostupnog Web API-ja, omogućava CRUD operacije klijentima za informacije o letovima za koje su zainteresovani, odnosno za rezervaciju istih.
- Rezervacija letova je simulirana lokalno, podaci se preko NATS broker-a prenose Gateway service-u koji šalje HTTP zahtev na endpoint Internal API-a.
- U okviru Internal API-ja se šalje zahtev za skladištenje informacija o rezervaciji u okviru MongoDB-a i klijent dobija povratnim putem odgovor.
- Weather service čita podatke iz dataset-a, šalje ih Gateway service-u preko broker-a, koji ih prosleđuje na Internal API endpoint.
- Gateway service takođe analizira podatke o vremenskoj prognozi i ukoliko su uslovi nepogodni, šalje zahtev na drugi Internal API endpoint za otkazivanje letova zbog nepogodnih uslova.
- Internal API tada vrši upit ka bazi i ukoliko za odgovarajući datum ima rezervisanih letova, otkazuje ih i upisuje informacije u Notifications kolekciju u okviru MongoDB-a, što bi trebalo da simulira obaveštenje odgovarajućim klijentima.

PROJEKAT I – SWAGGER UI

 Swagger
Powered by SMARTBEAR

Explore

Flight bookings API 1.0 OAS3

doc/openapi.yaml

This API is used to get flight booking information from Amadeus public API and also book flight tickets online.

Servers

http://{hostname}:{port}/api

▼

Computed URL: http://localhost:3000/api

Server variables

hostname:

localhost

 ▼

port:

3000

flights ^

GET

/flights/cities

Get city's code

▼

GET

/flights/airports-keyword

Get airports

▼

GET

/flights/airports-location

Get airports near location

▼

GET

/flights/destinations

Get destinations

▼

GET

/flights/dates

Get flight dates

▼

GET

/flights/offers

Get flight offers

▼

GET

/flights/airlines

Get airlines

▼

GET

/flights/tickets/{ticketId}

Get ticket

▼

DELETE

/flights/tickets/{ticketId}

Delete a booking

▼

GET

/flights/myTickets/{username}

Get tickets for a user

▼

POST

/flights/tickets

Book a ticket

▼

PATCH

/flights/tickets

Change ticket

▼

GET

/flights/notifications/{username}

Get notifications

▼

search ^

GET

/flights/cities

Get city's code

▼

GET

/flights/airports-keyword

Get airports

▼

GET

/flights/offers

Get flight offers

▼

GET

/flights/airlines

Get airlines

▼

cities ^

GET

/flights/cities

Get city's code

▼

airports ^

GET

/flights/airports-keyword

Get airports

▼

GET

/flights/airports-location

Get airports near location

▼

destinations ^

GET

/flights/destinations

Get destinations

▼

dates ^

GET

/flights/dates

Get flight dates

▼

offers ^

GET

/flights/offers

Get flight offers

▼

airlines ^

GET

/flights/airlines

Get airlines

▼

booking ^

GET

/flights/tickets/{ticketId}

Get ticket

▼

DELETE

/flights/tickets/{ticketId}

Delete a booking

▼

GET

/flights/myTickets/{username}

Get tickets for a user

▼

POST

/flights/tickets

Book a ticket

▼

PATCH

/flights/tickets

Change ticket

▼

notifications ^

GET

/flights/notifications/{username}

Get notifications

▼

weather ^

GET

/weather/forecast/{startDate}/{endDate}

Get weather forecast

▼

PUT

/weather/start

Start weather service

▼

forecast ^

GET

/weather/forecast/{startDate}/{endDate}

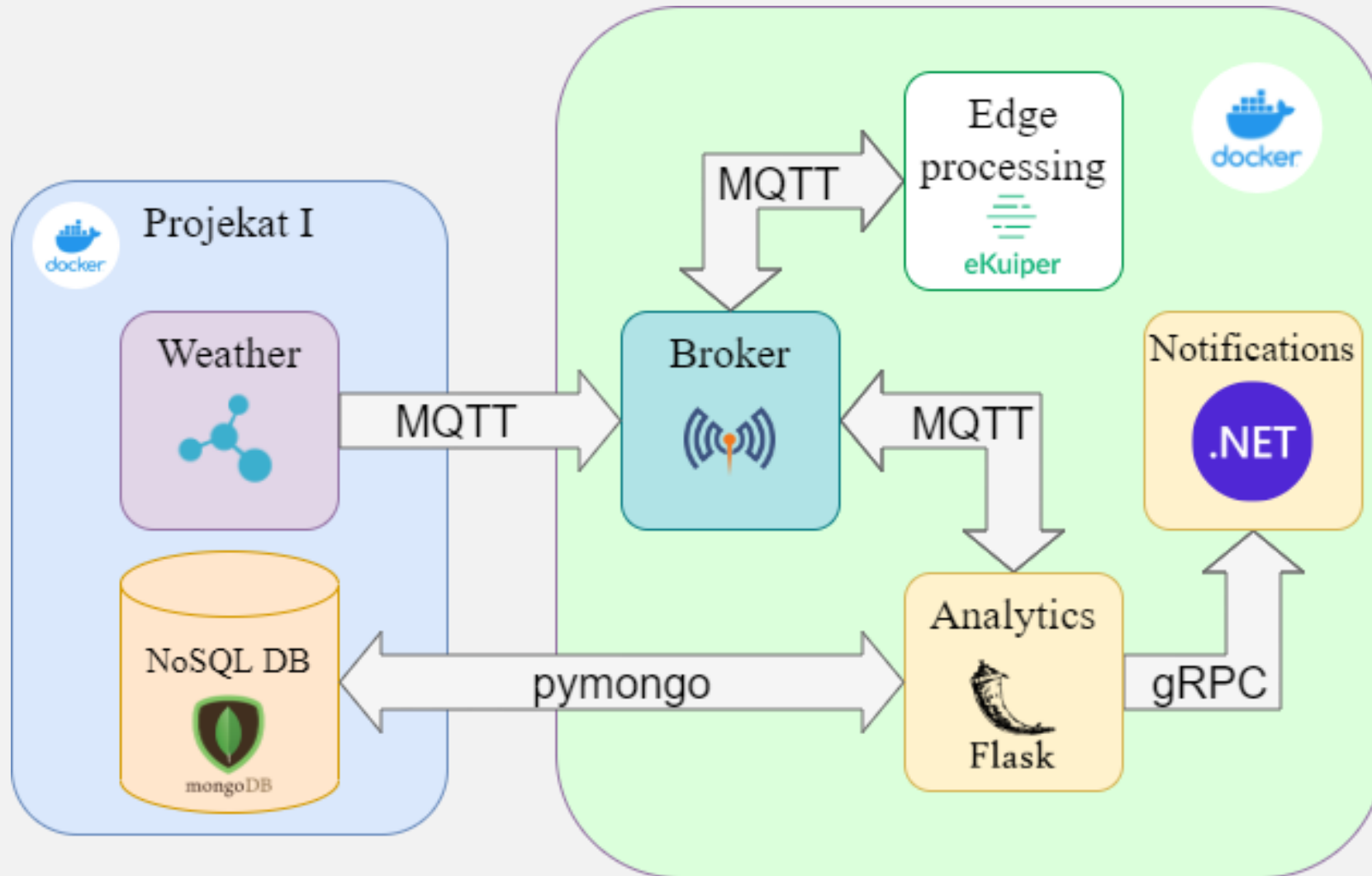
Get weather forecast

▼

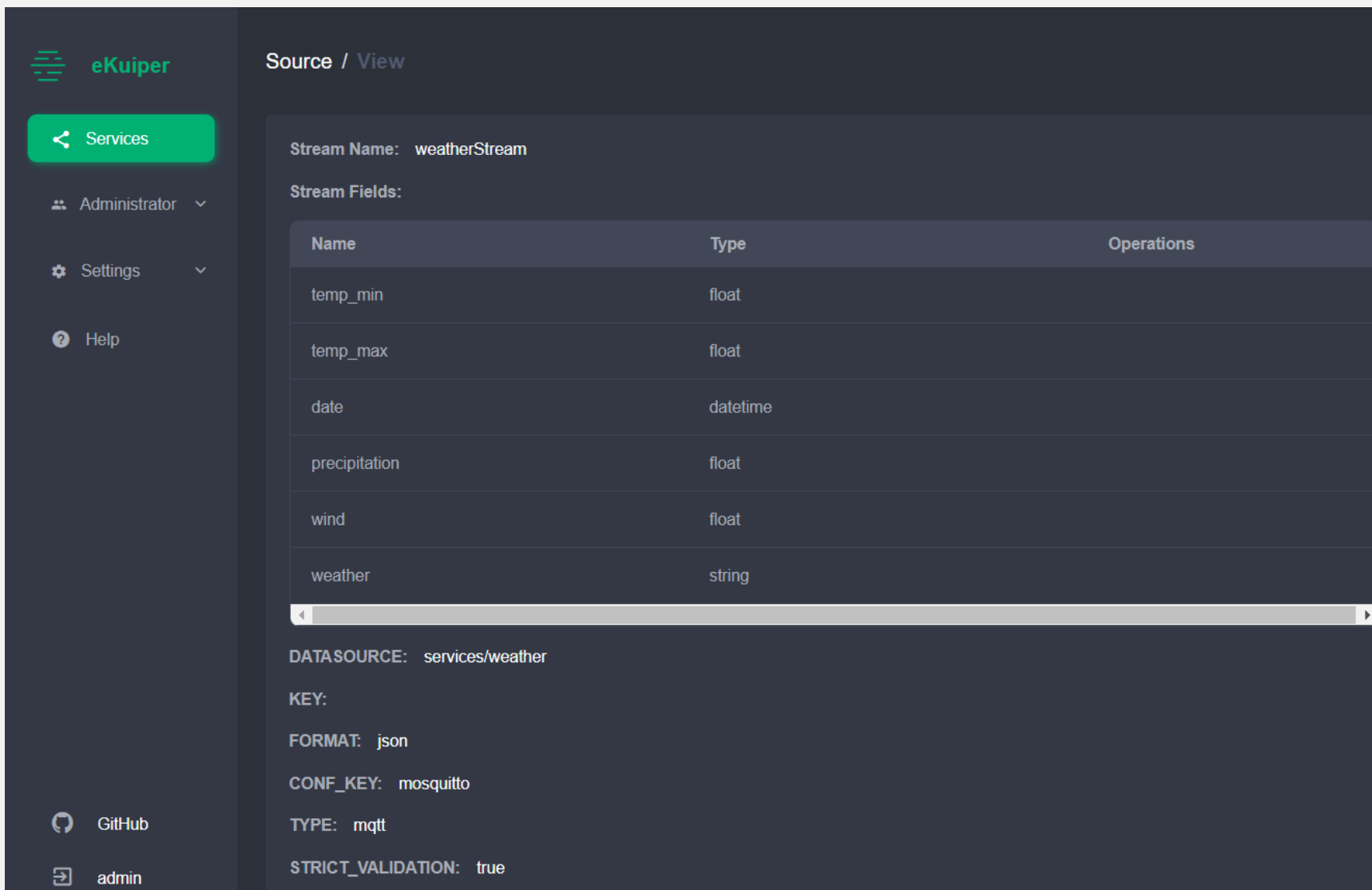
PROJEKAT II

- Izmena u okviru Projekta I, jeste publish koji vrši Weather service na Mosquitto MQTT broker.
- Podaci koji su emitovani kroz MQTT protokol odgovaraju strukturi podataka iz dataset-a Projekta I.
- Granični slučaj koji je odabran za okidanje akcije predstavlja vremenske uslove nepogodne za letenje.
- Po prijemu takve prognoze, pretplaćeni Analytics service šalje gRPC alert Notifications service-u.

PROJEKAT II - ARHITEKTURA



PROJEKAT II – EKUPIER FORMAT PODATAKA



The screenshot displays the eKuiper web interface. On the left is a dark sidebar with the eKuiper logo and navigation links: Services (highlighted in green), Administrator, Settings, and Help. At the bottom of the sidebar are links for GitHub and the user 'admin'. The main content area is titled 'Source / View' and shows the configuration for a stream named 'weatherStream'. It includes a table of stream fields with columns for Name, Type, and Operations. Below the table, several configuration parameters are listed: DATASOURCE, KEY, FORMAT, CONF_KEY, TYPE, and STRICT_VALIDATION.

eKuiper

Services

Administrator

Settings

Help

GitHub

admin

Source / View

Stream Name: weatherStream

Stream Fields:

Name	Type	Operations
temp_min	float	
temp_max	float	
date	datetime	
precipitation	float	
wind	float	
weather	string	

DATASOURCE: services/weather

KEY:

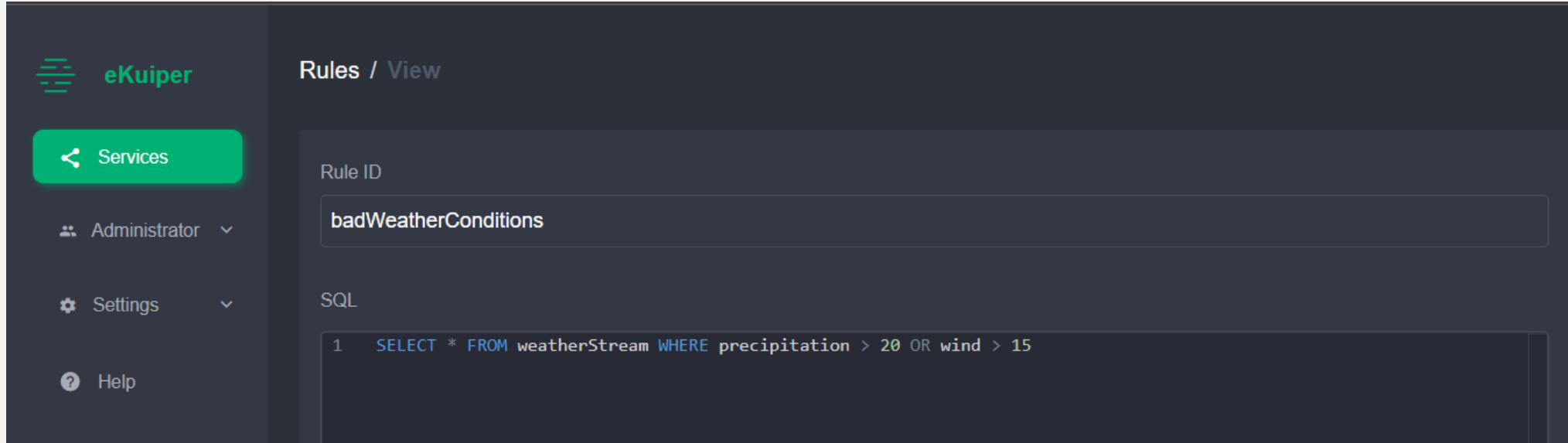
FORMAT: json

CONF_KEY: mosquitto

TYPE: mqtt

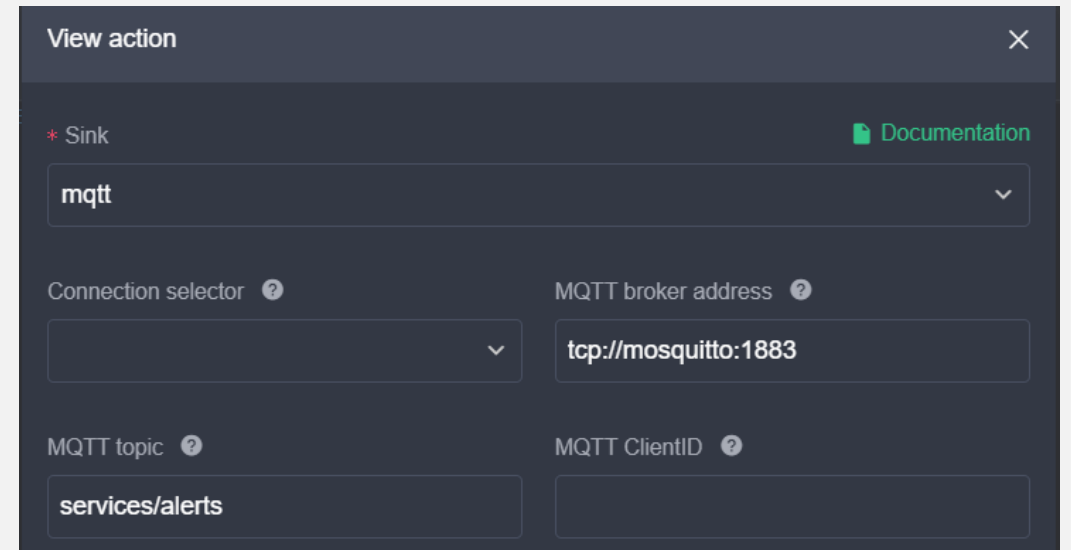
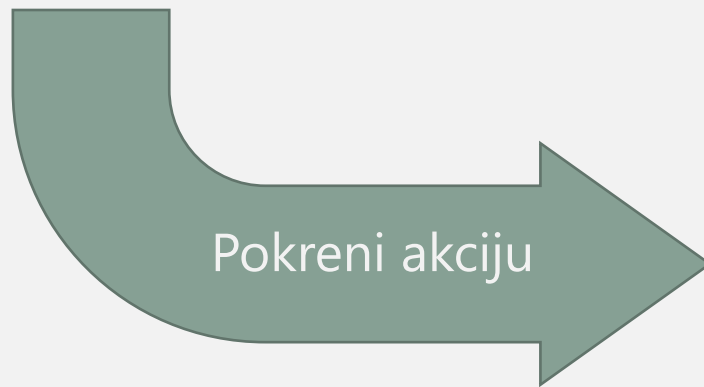
STRICT_VALIDATION: true

PROJEKAT II – EKUPIER PRAVILO I AKCIJA



The screenshot shows the eKuiper web interface. On the left is a dark sidebar with the eKuiper logo and navigation links: Services (highlighted in green), Administrator, Settings, and Help. The main area is titled 'Rules / View'. It contains a 'Rule ID' field with the value 'badWeatherConditions' and an 'SQL' field with the following query:

```
1 SELECT * FROM weatherStream WHERE precipitation > 20 OR wind > 15
```



The 'View action' dialog box is shown, featuring a close button (X) in the top right corner. It includes a 'Sink' dropdown menu with 'mqtt' selected and a 'Documentation' link. Below are four input fields with help icons (question marks):

- 'Connection selector' with a dropdown arrow.
- 'MQTT broker address' with the value 'tcp://mosquitto:1883'.
- 'MQTT topic' with the value 'services/alerts'.
- 'MQTT ClientID' which is currently empty.

PROJEKAT II – GRPC FORMAT PORUKA

```
syntax = "proto3";
```

```
option csharp_namespace = "Notifications.Protos";
```

Server implementiran u C#-u

```
package alerts;
```

Klijent implementiran u Python-u

```
service Alerts {  
    rpc Send(Alert) returns (Response) {}  
}
```

```
message Alert {  
    string id = 1;  
    string sender = 2;  
    string receiver = 3;  
    string date = 4;  
    string payload = 5;  
}
```

```
message Response {  
    string status = 1;  
}
```