User Guide

The goal of our system is to provide TU Delft employees with the ability to interface with a management system for a supercomputer called DelftBlue.

Docker setup

In order to use the system, you will first need to set up a docker container. Instructions for doing this are located in the file *kafka-example/README.md.*

Communication

Users can send the requests to the gateway running on port 8080.

Available endpoints:

- **/userView/regularUser** - enables users to see available resources per faculty for the next day - GET request
- **/userView/sysadmin** - enables sysadmins to see the booked resources, free resources, information on the nodes of the cluster and schedules jobs for a certain, with no limitations in the choosing of such - POST request
- **/clusterNode/add** - users can add a node to the supercomputer cluster - POST requests, as the body takes a *clusterNodeDTO*
- **/clusterNode/remove** - remove a node from the cluster - POST request, a the body takes a token representing the node
- **/faculty/pending** - gets the list of pending requests - POST request, takes a name of Faculty
- **/faculty/accept** - allows Faculty Reviewers to accept requests - POST request, as the body takes an *AcceptRequestDTO* with a list of IDs of requests to accept
- **/notification/{netId}** - gets notifications for the user with the given *netId* - GET request - checks if you are who you say you are
- **/user/add** - adds a new user - POST request, takes a *RequestDTO*
- **/user/authenticate** - authenticates an existing user, returning a *token* - POST request, takes *UserCredentials*
- **/request/new** - creates a new request and sends it to a faculty - POST request, takes a *requestDTO*

All requests, except for those connected to adding new users and authenticating them, must have an authentication token. Through this token, it is checked if the user has the necessary role to perform a given request and it is ensured that the users make requests only to their faculties and in their own name.

Example requests

Below are some example requests that can be run through the program Postman, to test the capabilities of our system. The examples are in order, so if they are executed on a running system, from top to bottom, they will work.
WARNING: if run out of order, these tests might give errors, for example because a user does not yet exist in the system.

- **/user/add**

(POST to localhost:8080/user/add)
First we will add a new user to the system, by executing the following example. After this, it is possible to authenticate the added user at the endpoint below.

```
{
  "netId": "zaidman",
  "password": "pass",
  "role": "EMPLOYEE",
  "faculties": ["EEMCS"]
}
```

- **/user/authenticate**

(POST to http://localhost:8080/user/authenticate)
Now that we created a user, let's authenticate them. This can be done using the following request. Copy the token you get as a response.

```
{
  "netId": "zaidman",
  "password": "pass"
}
```

- **/request/new**:

(POST to http://localhost:8080/request/new)
When sending requests, make sure to add the token you just copied to authorization > bearer token in the drop down menu > in the token field.

We now have an authenticated user in our system, so let's send a few requests. This first request has a preferred date quite far into the future, so this request will be added to the pending requests in the EEMCS faculty. This can be further tested by changing the faculty, random strings are not accepted.

```
{
  "name": "testName",
  "netId": "zaidman",
  "faculty": "EEMCS",
  "description": "desc",
  "preferredDate": "2030-02-03",
  "resource": {
    "cpu": 1,
    "gpu": 1,
    "memory": 1
  }
}
```

The next request is far into the past, and thus will not be accepted by the system.

```
{
  "name": "testName",
  "netId": "zaidman",
  "faculty": "EEMCS",
  "description": "desc",
  "preferredDate": "2000-02-03",
  "resource": {
    "cpu": 1,
    "gpu": 1,
    "memory": 1
  }
}
```

This request has more gpu and memory than cpu, which means that the request will not be accepted. To test this further, change the values for cpu, gpu and memory. The system denies requests for which cpu >= max(gpu, memory) does not hold.

```
{
  "name": "testName",
  "netId": "zaidman",
  "faculty": "EEMCS",
  "description": "desc",
  "preferredDate": "2023-02-03",
  "resource": {
    "cpu": 1,
    "gpu": 10,
    "memory": 10
  }
}
```

A last thing that could be tried with this endpoint is that 6 hours before the prefferedDate attribute, the request should be automatically accepted, and 5 minutes before that date, the request will be denied. However, since this is highly dependent on the day the request is made, we cannot give a pre-made example of this.

- **/faculty/pending**

Once we add a valid request to our system, we can retrieve it with the following example.

```
{
  "facultyName": "EEMCS"
}
```

- **/user/add**

(POST to localhost:8080/user/add)
We will add another user with the role of reviewer, to accept our current request.

```
{
  "netId": "reviewer",
  "password": "pass",
  "role": "FACULTY_REVIEWER",
  "faculties": [
    "EEMCS"
  ]
}
```

- **/user/authenticate**

(POST to http://localhost:8080/user/authenticate)
We authenticate the reviewer. Again make sure to copy the token.

```
{
  "netId": "reviewer",
  "password": "pass"
}
```

- **/faculty/accept**

(POST to http://localhost:8080/faculty/accept)
We can accept the running request with this example. Make sure to add the token in the same place as when sending the request.

```
{
  "facultyName": "EEMCS",
  "requests": [
    1
  ]
}
```

- **/userView/regularUser**

(GET to http://localhost:8080/userView/regularUser)

Any user can target this endpoint to obtain information of the available resources for the following day regarding the faculties they belong to.

- **/userView/sysadmin**

(POST to http://localhost:8080/userView/sysadmin)

Only sysadmins can target this endpoint - any other user type will receive status code 401. For you to use this endpoint correctly, you must send a json object in the following format:

```
{
  "year”: 2022,
  “month”: 12,
  “day”: 25
}
```

From this request you will receive a view on the number of available resources for every faculty, the number of available resources on the whole system, information on the capacity of every node of the system, and all the schedules jobs from all faculties for the day you have specified.

- **/clusterNode/add**

(POST to localhost:8080/clusterNode/add)
The user can also add their own nodes to the cluster.

```
{
  "token":{
    "tokenValue": "testTokenValue"
  },
  "ownerName":{
    "name": "zaidman"
  },
  "url":{
    "urlValue": "testUrlValue"
  },
  "resources":{
    "cpu": "500",
    "gpu": "300",
    "memory": "300"
  }
}
```

The capacity of these nodes counts towards the freepool. Note that you can only add nodes with your name, hence why this will fail:

```
{
  "token":{
    "tokenValue": "testTokenValue"
  },
  "ownerName":{
    "name": "notzaidman"
  },
  "url":{
    "urlValue": "testUrlValue"
  },
  "resources":{
    "cpu": "3",
    "gpu": "2",
    "memory": "1"
  }
}
```

- **/clusterNode/remove**

(POST to localhost:8080/clusterNode/remove)

While this endpoint is not yet fully functional, user can also delete their nodes:

```
{
  "tokenValue": "testTokenValue"
}
```