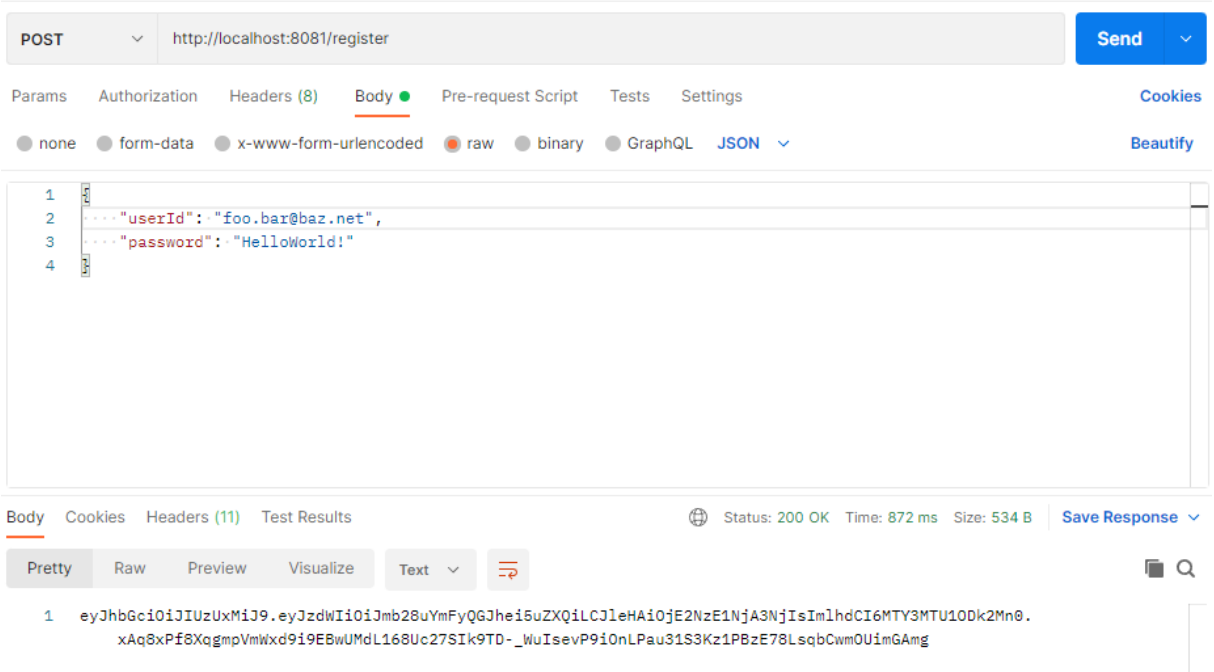
# Integration testing with Postman

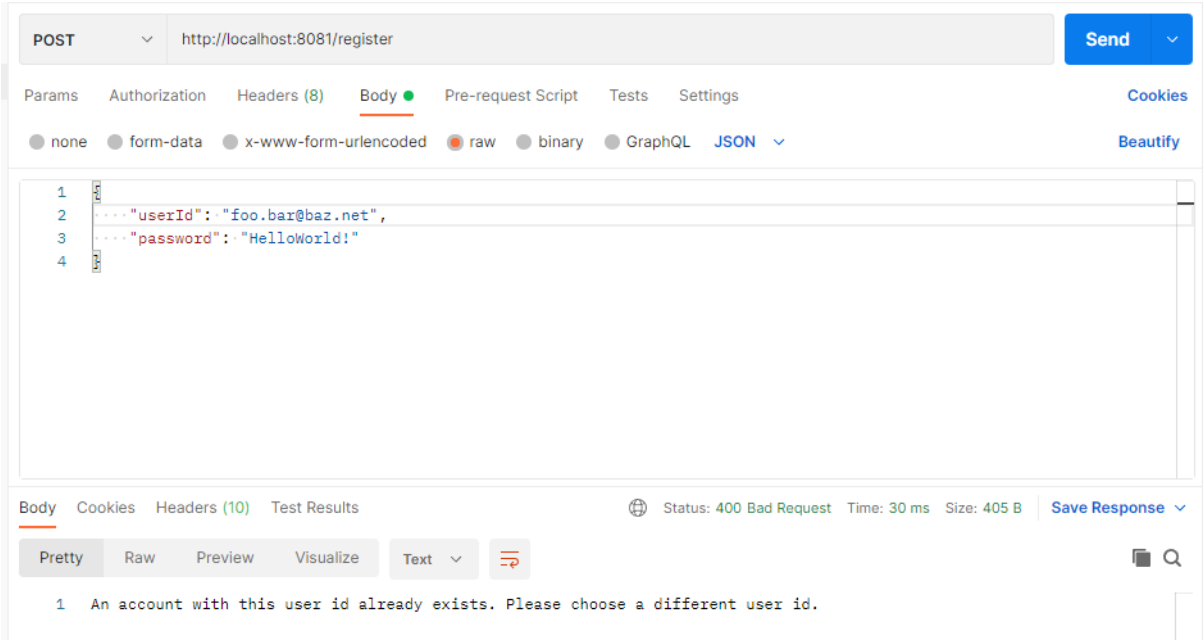
Besides with written tests in Java, the microservices and their interactions have also been tested in a more practical setting using http requests sent with Postman. This document includes screenshots of Postman requests showing that the overall behaviour of the system is indeed as expected.

## Authentication microservice

The registration path will respond with a token when the credentials are acceptable.




If the credentials are not in the correct format or an account with the specific username already exists, it will return an error informing the client of what they did wrong. Below are a couple examples of different error messages (there are more).



POST http://localhost:8081/register Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "userId": "foo.bar''''@baz.net",
    "password": "HelloWorld!"
}
```

Body Cookies Headers (10) Test Results Status: 400 Bad Request Time: 7 ms Size: 459 B Save Response

Pretty Raw Preview Visualize Text

```
Your user id contains illegal characters. Please only use letters, numbers and the following characters: !#$%&()*+,-./:;<=>?@^_`{|}~
```

POST http://localhost:8081/register Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "userId": "foo.bar",
    "password": "HelloWorld!"
}
```

Body Cookies Headers (10) Test Results Status: 400 Bad Request Time: 14 ms Size: 364 B Save Response

Pretty Raw Preview Visualize Text

```
Your user id must be an email address.
```

If the client already has an existing account, they can use the /authenticate path. This will also return errors when the format or credentials are incorrect. The client will again receive a token when the credentials are correct.

POST http://localhost:8081/authenticate ... Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2     "userId": "foo.bar@baz.net",
3     "password": "HelloWorld!"
4 }
```

Body Cookies Headers (11) Test Results Status: 401 Unauthorized Time: 275 ms Size: 403 B Save Response

Pretty Raw Preview Visualize Text

```
1 UserId or password incorrect. Please try again.
```

POST http://localhost:8081/authenticate ... Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2     "userId": "foo.bar@baz.net",
3     "password": "HelloWorld!"
4 }
```

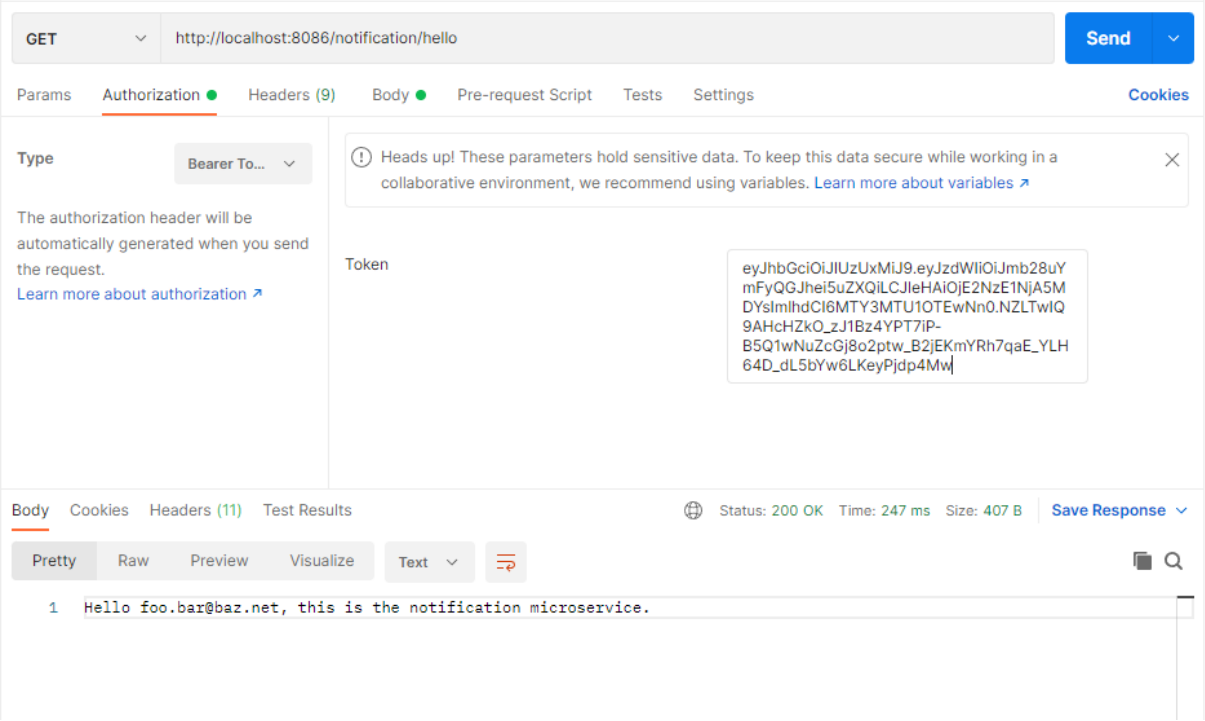
Body Cookies Headers (11) Test Results Status: 200 OK Time: 341 ms Size: 534 B Save Response

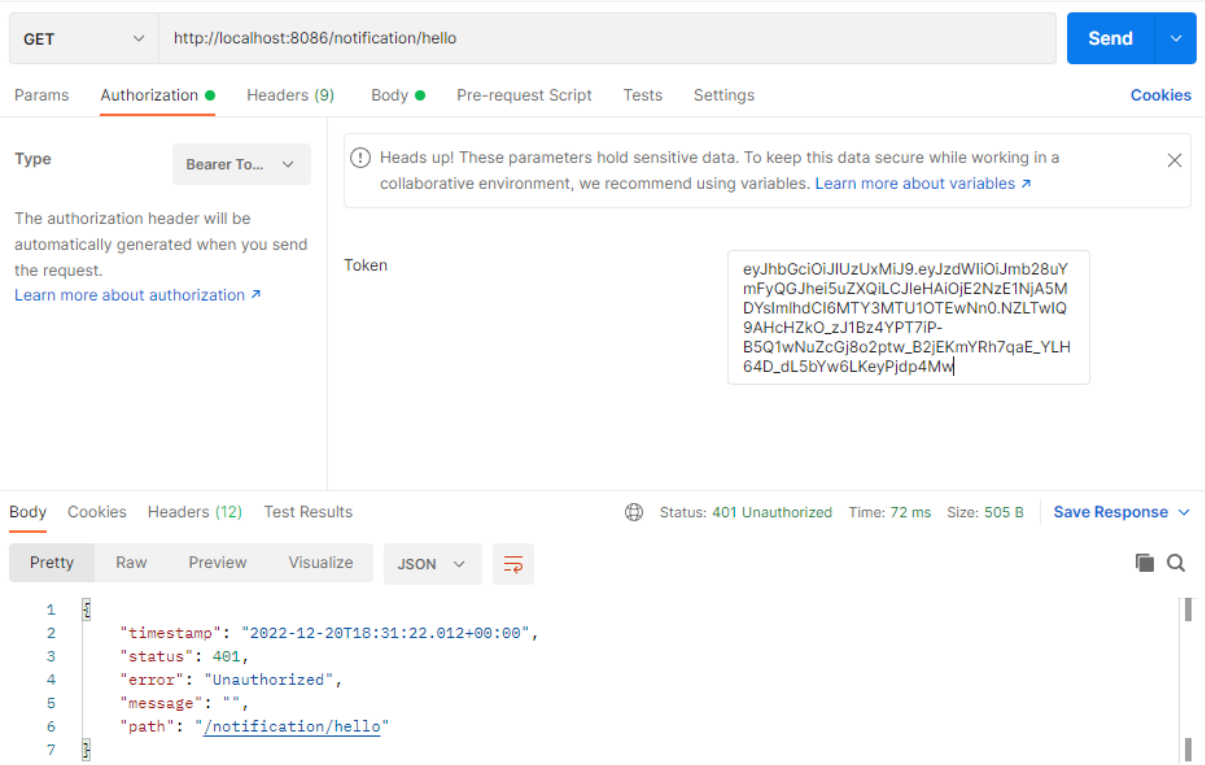
Pretty Raw Preview Visualize Text


```
1 eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJmb28uYmFyQGJhei5uZXQiLCJleHAiOjE2NzE1NjA5MDYsImlhdCI6MTY3MTU1OTEwNn0.
    NZLTwIQ9AHcHZkO_zJ1Bz4YPT7iP-B5Q1wNuZcGj8o2ptw_B2jEKmYRh7qaE_YLH64D_dL5bYw6LKeyPjdp4Mw
```


When the clients have correctly authenticated themselves or registered, they can make requests to the other microservices with the token as authorization. The other microservices all use the same authentication package to handle verification of the token. This package also provides the getUserId method, which the services can use to extract the userId from the token.

For example, below, during a hello request to the notification service, it will extract the UserId and use it in its response. Also included is a response from the same service with an expired token in the request. The token will expire after half an hour.

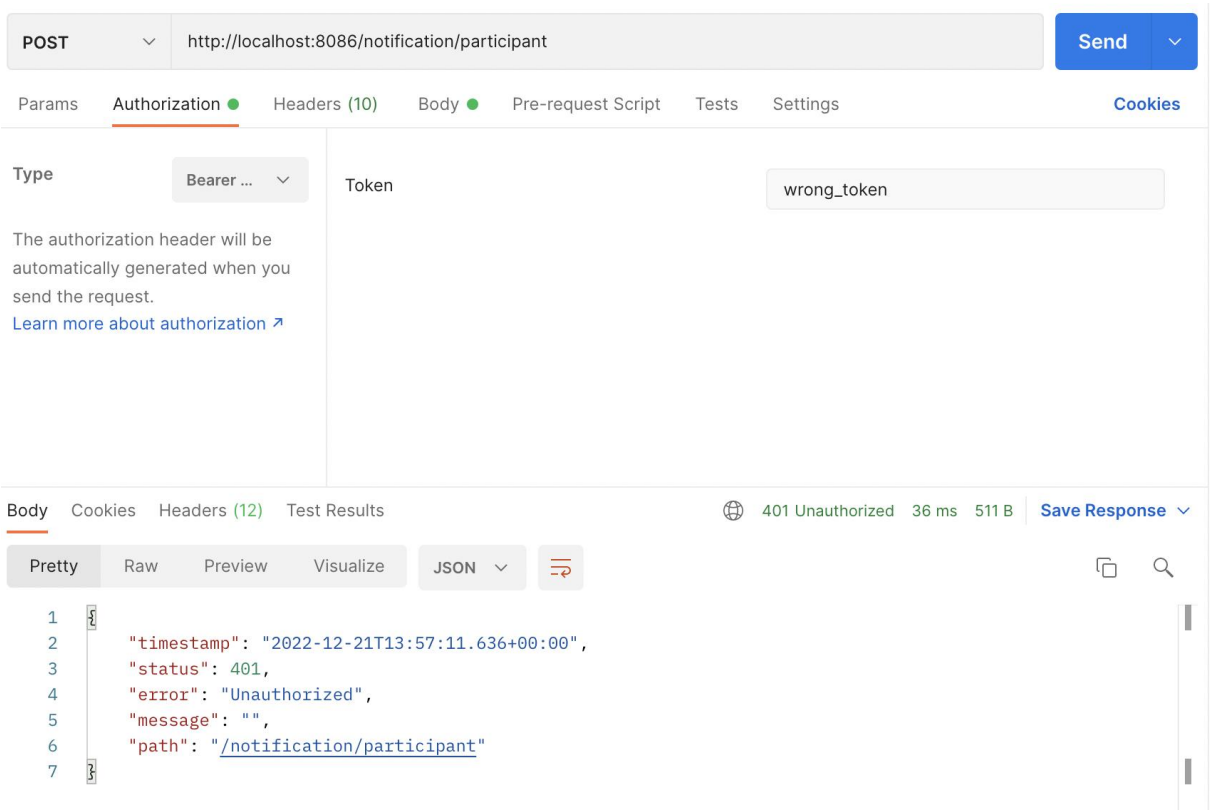



The authentication package is the same in all four other microservices. So, the demonstrated token authentication and id extraction work on all the services.

# Notification

In order to send notifications via email, one must be authenticated first. Otherwise an error will be shown, mentioning this has not been done properly. After following the steps mentioned in the Authentication section, include your bearer token in the "Authorization" page in Postman for each request you want to make.




To email a rower regarding a decision about his participation in a training / competition, one should include the following details in the request body: the id of the participant, the id of the activity, the timeslot he applied for and the decision from the owner of the activity:

POST http://localhost:8086/notification/participant Send

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

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "participantId": "user@gmail.com",
    "activityId": 777,
    "timeslot": {
        "startTime": "2007-12-03T10:15:30.332",
        "endTime": "2008-12-03T10:15:30.332"
    },
    "decision": true
}
```

Body Cookies Headers (11) Test Results 200 OK 2.29 s 362 B Save Response

Pretty Raw Preview Visualize JSON

```
Email sent successfully.
```

To email a rower regarding a modification or the deletion of an activity he has been matched with, the following request should be made mentioning the participant id, the activity id and the timeslot:

POST http://localhost:8086/notification/activity-changed Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "participantId": "user@gmail.com",
    "activityId": 999,
    "timeslot": {
        "startTime": "2007-12-03T10:15:30.332",
        "endTime": "2008-12-03T10:15:30.332"
    }
}
```

Body Cookies Headers (11) Test Results 200 OK 1429 ms 362 B Save Response

Pretty Raw Preview Visualize JSON

```
Email sent successfully.
```

To email a publisher of an activity when a rower requested to participate in that activity at a certain timeslot, the following request should be made:

POST http://localhost:8086/notification/publisher Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "ownerId": "owner@gmail.com",
    "participantId": "user@gmail.com",
    "activityId": 777,
    "timeslot": {
        "startTime": "2007-12-03T10:15:30.332",
        "endTime": "2008-12-03T10:15:30.332"
    }
}
```

Body Cookies Headers (11) Test Results 200 OK 1580 ms 362 B Save Response

Pretty Raw Preview Visualize JSON

```
Email sent successfully.
```

This is an example of email you can receive as a rower, when you get accepted for an activity:

New notification regarding rowing competitions Inbox ×

**rowing.competitions@gmail.com** Tue, Dec 20, 10:55 PM (17 hours ago)

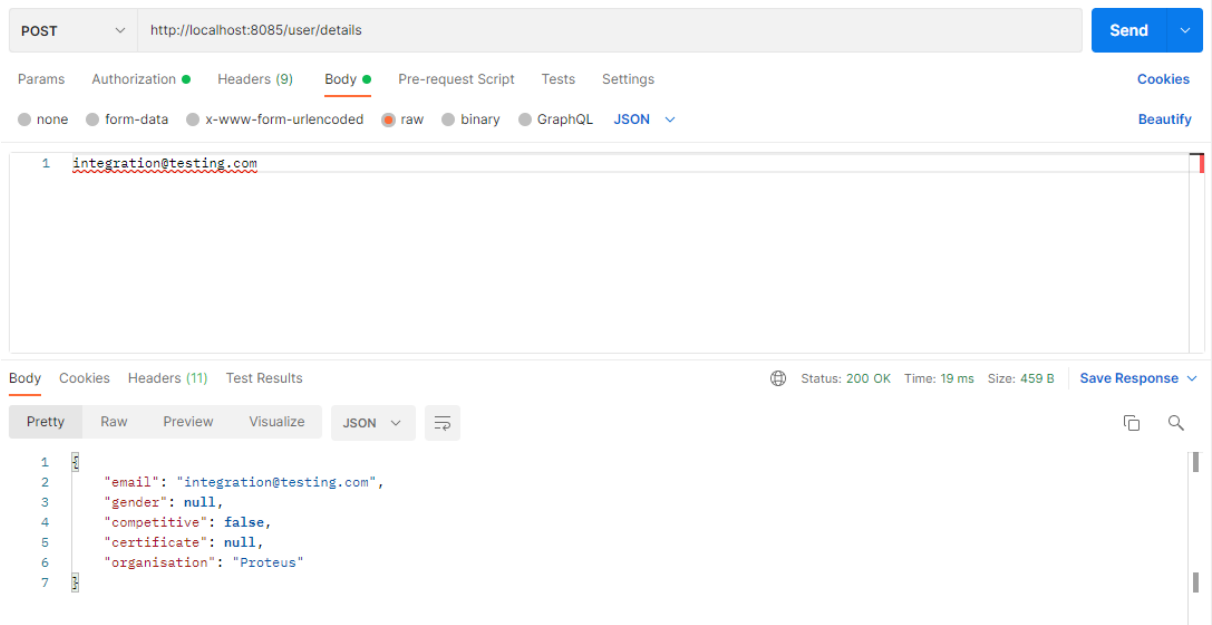
to me

Congratulations! You have been accepted for activity 777. You are expected to be there between 2007-12-03T10:15:30.332 - 2008-12-03T10:15:30.332.

Reply Forward

# Users

We can use postman to verify that the interaction with other micro-services is as desired. Additionally, to the client-based functionality displayed in the README, the User micro-service also supplies an endpoint for the Matching micro-service to obtain all the user data associated with an email. We can test this endpoint by sending a request identical to the form of request the matching micro-service would send.




The only request sent to other micro-services from the User micro-service is a validation request for the certificate. We verify certificates in this way as they are supplied by clients for both user data and activity data, and having a common verification point increases maintainability. As the certificates are rejected/accepted successfully we have tested the communication between the micro-services.

POST http://localhost:8085/user/updatemydata Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "email": "integration@testing.com",
    "certificate": "trustme"
}
```

Body Cookies Headers (11) Test Results Status: 409 Conflict Time: 794 ms Size: 401 B Save Response

Pretty Raw Preview Visualize Text

```
This certificate is not recognized by our system.
```

POST http://localhost:8085/user/updatemydata Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "email": "integration@testing.com",
    "certificate": "4+"
}
```

Body Cookies Headers (11) Test Results Status: 200 OK Time: 50 ms Size: 459 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "email": "integration@testing.com",
    "gender": null,
    "competitive": false,
    "certificate": "4+",
    "organisation": "Proteus"
}
```

There has also been done integration testing of the components within the Users micro-service through postman. The flow between the components in the micro-service can be verified in the postman requests both in this file and additionally in the requests displayed in the README file. These tests are all in addition to the test cases included in the code.

We have not included screenshots of every type of validation error as there are many but will include another example and have tested all other type of validation errors in postman.

POST http://localhost:8085/user/updatemydata Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2     "email": "integration@testing.com",
3     "gender": "notAGender"
4 }
```

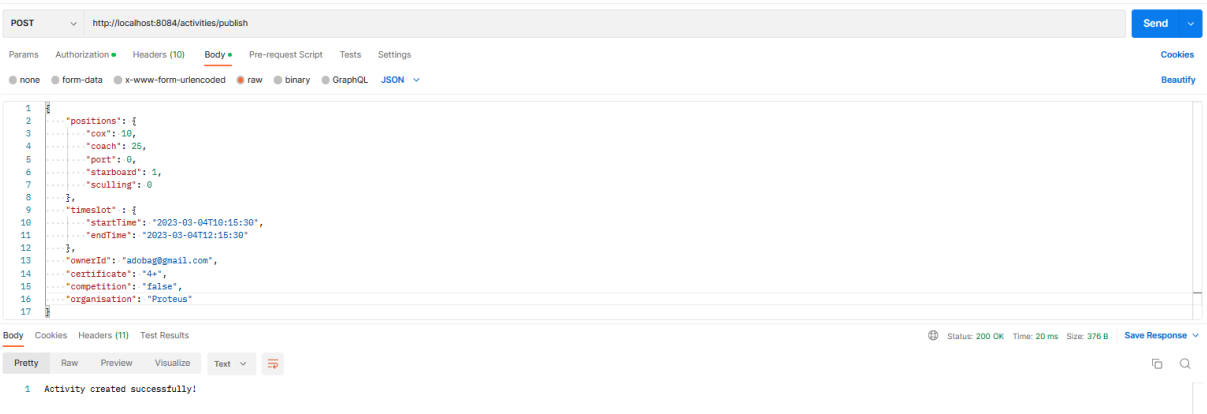
Body Cookies Headers (11) Test Results Status: 409 Conflict Time: 23 ms Size: 398 B Save Response

Pretty Raw Preview Visualize Text

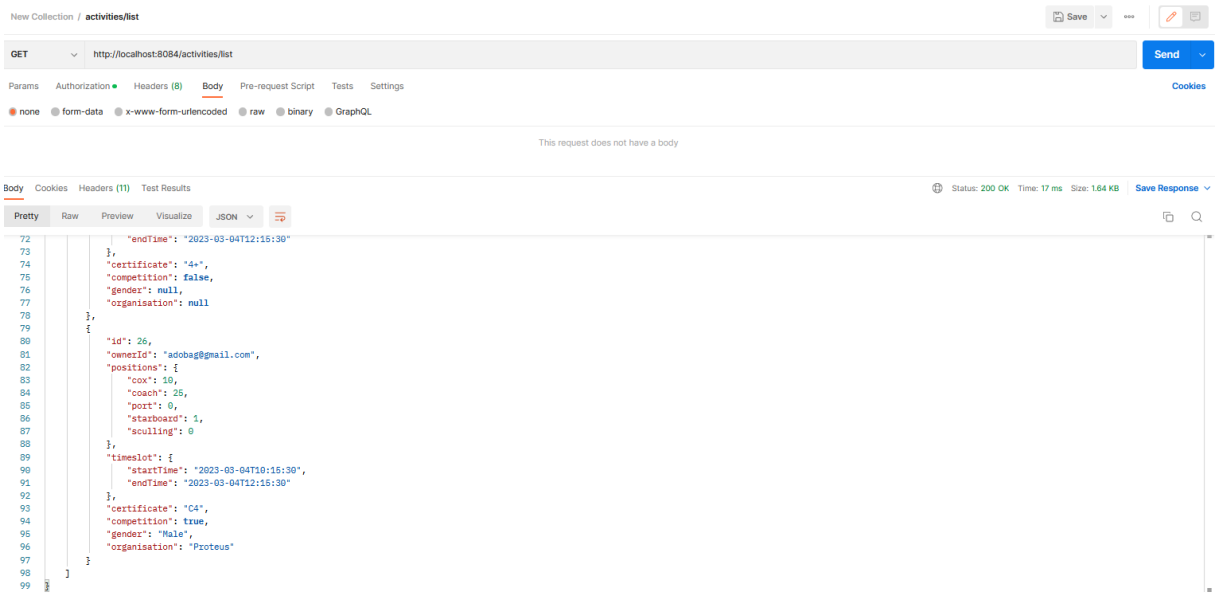
```
1 Please enter your gender as 'Male' or 'Female'
```

# Activities

To test the activities micro-service, the tester should create some activities. The removal of some fields that are needed will cause a bad request, which is the expected behaviour. Meanwhile fields, such as organisation, competition and gender are only required for competitions and not regular trainings.




If the tester authorised as the owner, they will be able to see those activities using the “list” endpoint.




Or the ones created by any user using the “within-timeslot” endpoint, where you can change some values to make sure activities are included or not included depending on their timeslots.

New Collection / activities/within-timeslot

POST http://localhost:8084/activities/within-timeslot Send

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

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "startTime": "2022-12-24T10:12:30",
    "endTime": "2024-03-04T20:15:30"
}
```

Body Cookies Headers (11) Test Results Status: 200 OK Time: 20 ms Size: 1.37 KB Save Response

Pretty Raw Preview Visualize JSON


```
{
    "activities": [
        {
            "id": 16,
            "ownerId": "adobag1@gmail.com",
            "positions": {
                "cox": 1,
                "coach": 0,
                "port": 8,
                "starboard": 0,
                "sculling": 0
            },
            "timeslot": {
                "startTime": "2023-03-04T10:15:30",
                "endTime": "2023-03-04T12:15:30"
            },
            "certificate": "4+",
            "competition": true
```


There is an endpoint for getting a timeslot by activity id.

GET http://localhost:8084/activities/26 Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (11) Test Results Status: 200 OK Time: 13 ms Size: 413 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "startTime": "2023-03-04T10:15:30",
    "endTime": "2023-03-04T12:15:30"
}
```


There is also the "edit" endpoint, used for editing the activities, which should also be tested.

PATCH http://localhost:8084/activities/edit Send

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

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify


```
{
    "id": 26,
    "certificate": "C4",
    "gender": "Male"
}
```


Body Cookies Headers (11) Test Results Status: 200 OK Time: 36 ms Size: 612 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "id": 26,
    "ownerId": "adobag@gmail.com",
    "positions": {
        "cox": 10,
        "coach": 25,
        "port": 0,
        "starboard": 1,
        "sculling": 0
    },
    "timeslot": {
        "startTime": "2023-03-04T10:15:30",
        "endTime": "2023-03-04T12:15:30"
    },
    "certificate": "C4",
    "competition": true,
    "gender": "Male",
    "organisation": "Proteus"
}
```


The “update” endpoint decreases the selected position but does not allow the counts to go below zero.

PUT http://localhost:8084/activities/update/26 Send

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

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
coach
```

Body Cookies Headers (11) Test Results Status: 200 OK Time: 18 ms Size: 612 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "id": 26,
    "ownerId": "adobag@gmail.com",
    "positions": {
        "cox": 10,
        "coach": 24,
        "port": 0,
        "starboard": 1,
        "sculling": 0
    },
    "timeslot": {
        "startTime": "2023-03-04T10:15:30",
        "endTime": "2023-03-04T12:15:30"
    },
    "certificate": "C4",
    "competition": true,
    "gender": "Male",
    "organisation": "Proteus"
}
```


Finally, you can delete them with the following endpoint. It is recommended to try to delete an activity that was created by another test account to make sure it is not possible.

DELETE http://localhost:8084/activities/delete/21

Params Authorization Headers (8) Body Pre-request Script

Type Bearer Token

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize Text

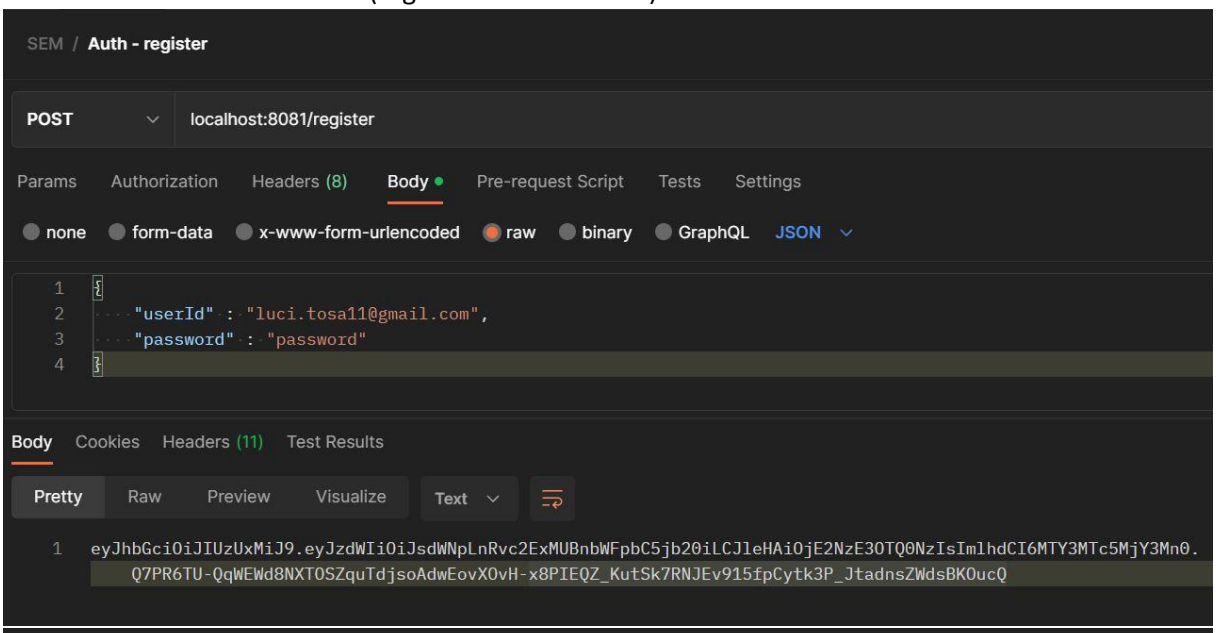
```
Activity with the id 21 has been deleted successfully!
```

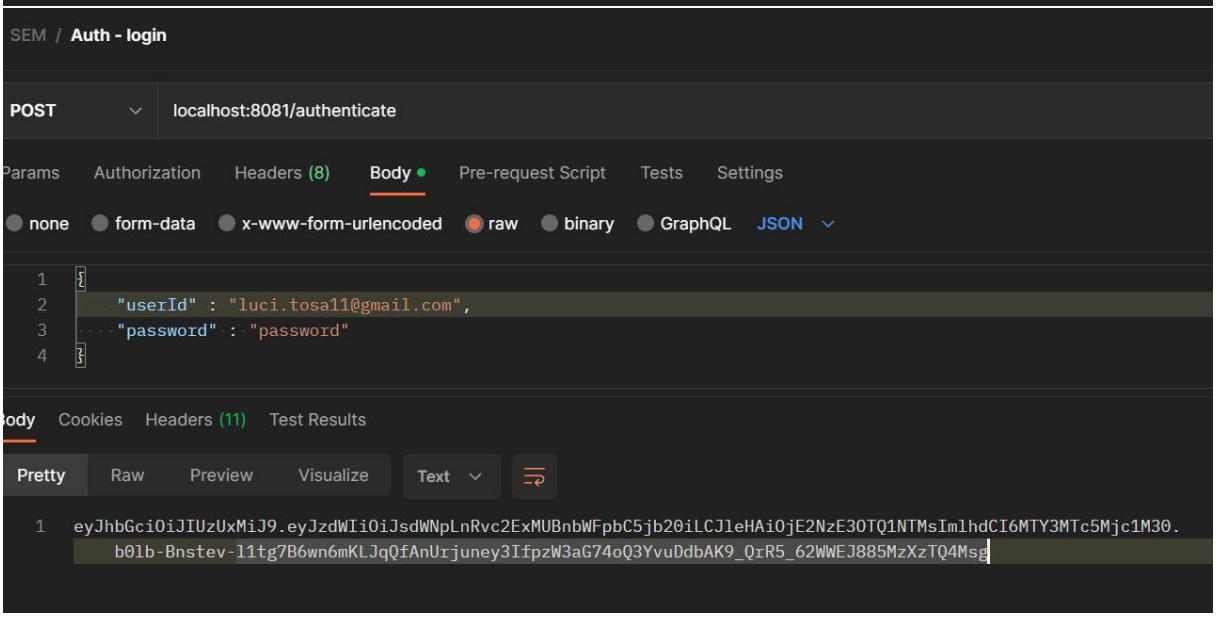
# Matching

There are several steps that the **'Matching' microservice** is responsible for in order to ensure the main functionality of the 'Rowing' app. That is why it is crucial for the entire flow of the application to be tested using Postman integration testing like shown in the following steps.

The first thing before starting to interact directly with the **'Matching' microservice** is to set up an account for a new user (steps 1. and 2.)

1. **'Authentication' microservice** (register + authenticate)



2. **'Users' microservice** (set up details of new user)

SEM / **Users - setup**

**POST** localhost:8085/user/newuser ...

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "email" : "luci.tosa11@gmail.com",
3     "gender" : "Male",
4     "competitive" : true,
5     "certificate" : "C4",
6     "organisation" : "Laga"
7 }
```

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "email": "luci.tosa11@gmail.com",
3     "gender": "Male",
4     "competitive": true,
5     "certificate": "C4",
6     "organisation": "Laga"
7 }
```

Then, in order to be able to match a user to an activity we need to have activities published in the first place and for that we will make use of the corresponding subsystem.

3. **'Activity' microservice** (publish a new activity - to be used for matching the user)

- Publish a Training

SEM / **Activity - publish**

**POST** localhost:8084/activities/publish

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

```
{
    "ownerId" : "luci.tosa11@gmail.com",
    "positions" : {
        "cox" : 1,
        "coach" : 1,
        "port" : 0,
        "sculling" : 0,
        "starboard" : 0
    },
    "timeslot" : {
        "startTime" : "2023-03-04T10:12",
        "endTime" : "2023-03-04T10:30"
    },
    "certificate" : "4+",
    "competition" : false
}
```

Body | Cookies | Headers (11) | Test Results

Pretty | Raw | Preview | Visualize | Text

```
Activity created successfully!
```

- Publish a Competition

SEM / **Activity - publish**

**POST** localhost:8084/activities/publish

Params | Authorization | Headers (9) | **Body** | Pre-request Script

none | form-data | x-www-form-urlencoded | raw | binary

```
2    "ownerId" : "luci.tosali@gmail.com",
3    "positions" : {
4        "cox" : 1,
5        "coach" : 1,
6        "port" : 0,
7        "sculling" : 0,
8        "starboard" : 0
9    },
10   "timeslot" : {
11       "startTime" : "2023-03-04T10:12",
12       "endTime" : "2023-03-04T10:30"
13   },
14   "certificate" : "4+",
15   "competition" : false,
16   "gender" : "Female",
17   "organisation" : "Laga"
18 }
```

Body | Cookies | Headers (11) | Test Results

Pretty | Raw | Preview | Visualize | Text

```
1  Activity created successfully!
```

After completing all this steps we repeat steps 1. and 2. with another email to create the user that will act as the participant to the activities created by the previously created user.

4. **'Matching' microservice**

- The user submits the availability and the desired position and it gets back a list of possible activities that they are able to participate to given their profile, timeslot and position chosen (inside this request the filtering of activities is done based on the constraints given and ensured by the Chain of Responsibility pattern - note that all cases for filtering are tested by Unit testing inside the app)

SEM / **Matching - submit**

**POST** localhost:8083/matching/submit ...

Params Authorization Headers (9) Body Pre-request Script Tests Setting

none form-data x-www-form-urlencoded raw binary GraphQL JSO

```
{
    "timeslot" : {
        "startTime" : "2023-03-04T06:12",
        "endTime" : "2023-03-04T16:30"
    },
    "position" : "coach"
}
```

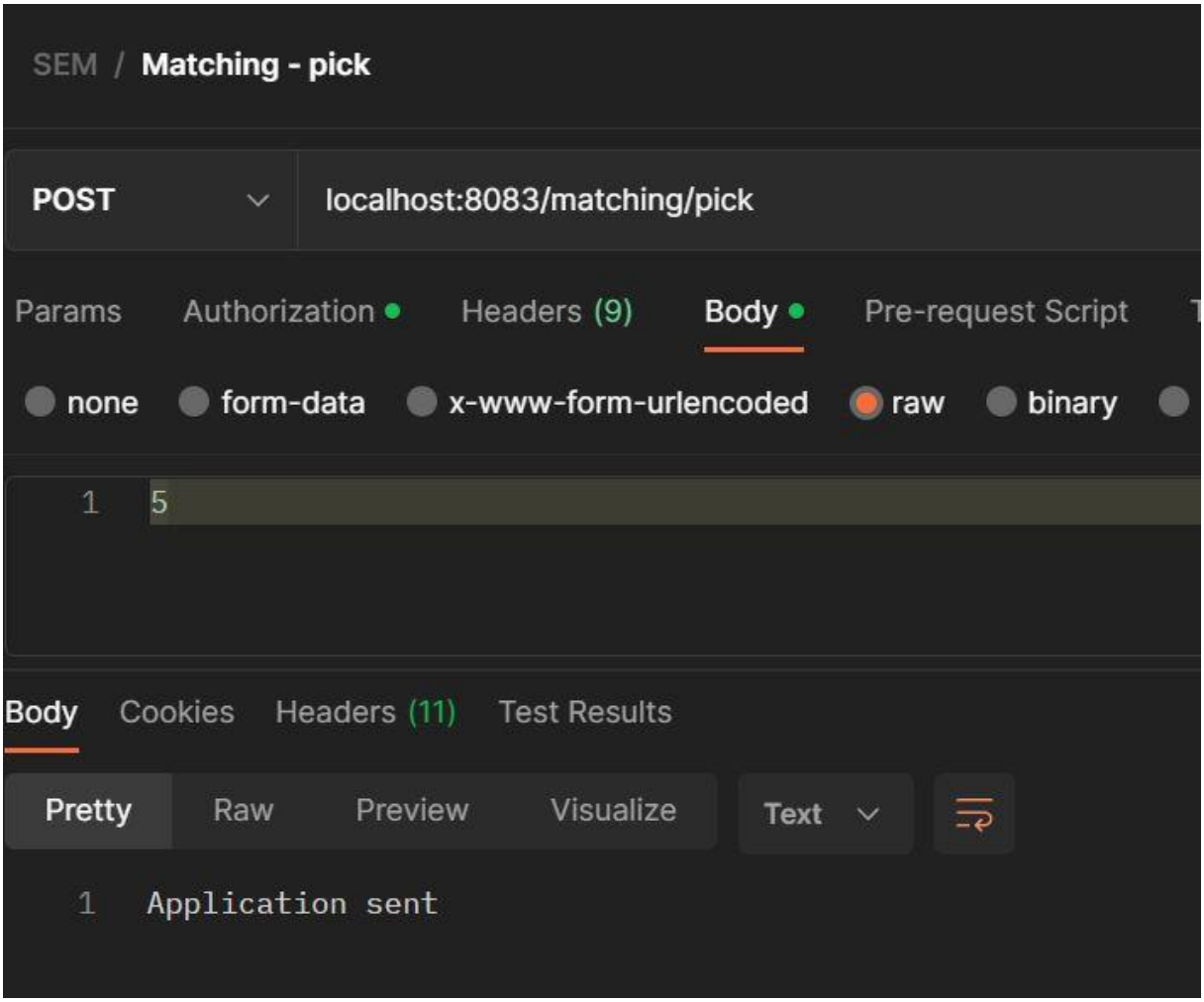
Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

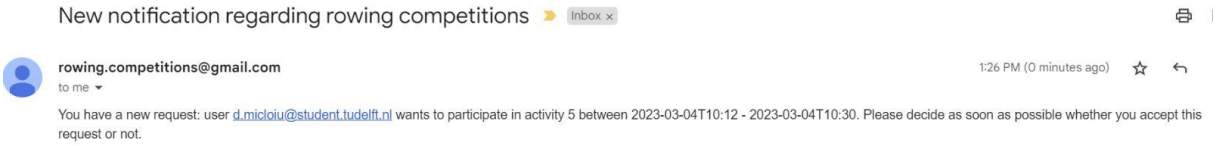
```
{
    "activities": [
        {
            "matchId": 4,
            "type": "TRAINING",
            "timeslot": {
                "startTime": "2023-03-04T10:12:00",
                "endTime": "2023-03-04T10:30:00"
            }
        },
        {
            "matchId": 5,
            "type": "COMPETITION",
            "timeslot": {
                "startTime": "2023-03-04T10:12:00",
                "endTime": "2023-03-04T10:30:00"
            }
        }
    ]
}
```

-

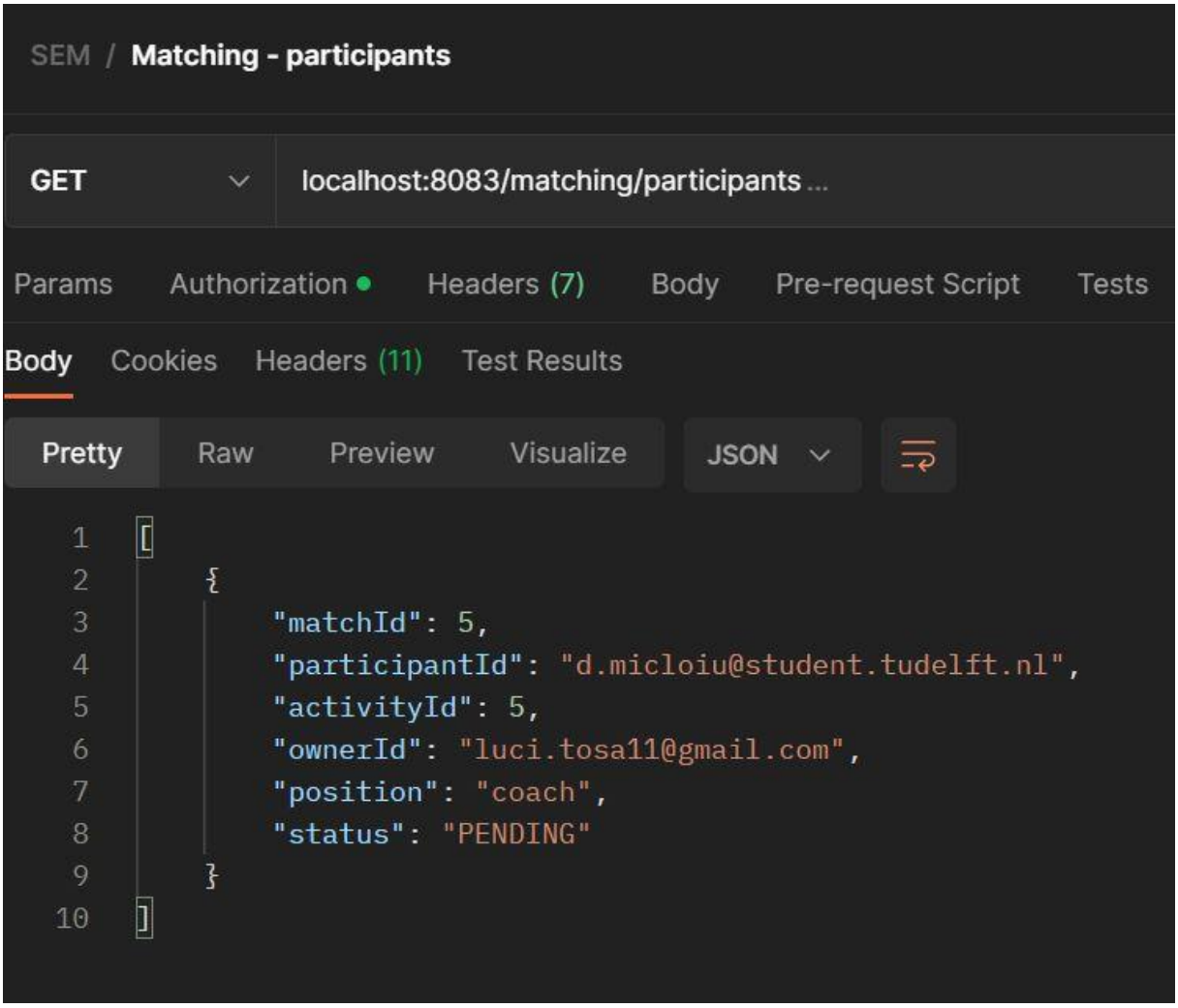
- The user then picks an activity they want to participate to by doing a request with the id of the match (the status of the match is now changed to Pending)




- When doing this, by communicating with the **'Notification' microservice**, the owner of the activity gets the following email



- Next, based on a "pull" manner, the owner can log in into the app and request all the participants that have been matched to their activity and also see the status they are in: PENDING, MATCHED, ACCEPTED, DECLINED




- Following this, the owner does a request to the service to decide whether they accept or decline the user by sending the id of the match and a boolean value (TRUE - accepted / FALSE - declined).

1. The user is accepted

SEM / **Matching - decide**

**POST** localhost:8083/matching/decide ...

Params Authorization Headers (9) Body Pre-request Script Te

none form-data x-www-form-urlencoded raw binary G

```
1 {
2     "matchId" : 5,
3     "decision" : true
4 }
```

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize Text

```
1 Notification sent successfully !
```

Gets an email telling them they have been accepted

**New notification regarding rowing competitions**

R rowing.competitions@gmail.com <rowing.competitions@gmail.com>

To: Diana Micloiu

Congratulations! You have been accepted for activity 5. You are expected to be there between 2023-03-04T10:12 - 2023-03-04T10:30.

And a request is done internally to the **'Activities' microservice** for decreasing by one the available places for the activity the user has been accepted to and the position they requested.


```
{
    "id": 5,
    "ownerId": "luci.tosa11@gmail.com",
    "positions": {
        "cox": 1,
        "coach": 0,
        "port": 0,
        "starboard": 0,
        "sculling": 0
    },
```


Now we can do a request to see all the matches that have status matched and we will get only one since the other match has now status accepted

SEM / **Matching - applications**

**GET** localhost:8083/matching/match/MATCHED

Params | Authorization | Headers (7) | Body | Pre-request Script | Test

**Type** Bearer Token

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

Heads up! These p
variables. variables

**Token**

Body | Cookies | Headers (11) | Test Results

Pretty | Raw | Preview | Visualize | JSON


```
[
    {
        "matchId": 4,
        "participantId": "d.micloiu@student.tudelft.nl",
        "activityId": 4,
        "ownerId": "luci.tosa11@gmail.com",
        "position": "coach",
        "status": "MATCHED"
    }
]
```

SEM / **Matching - applications**

**GET** localhost:8083/matching/match/ACCEPTED

Params | **Authorization** | Headers (7) | Body | Pre-request Script | Tests

**Type** **Bearer Token**

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

Heads up! These para
variables. variables ↗

**Token**

Body | Cookies | Headers (11) | Test Results

**Pretty** | Raw | Preview | Visualize | **JSON**


```
[
    {
        "matchId": 5,
        "participantId": "d.micloiu@student.tudelft.nl",
        "activityId": 5,
        "ownerId": "luci.tosa11@gmail.com",
        "position": "coach",
        "status": "ACCEPTED"
    }
]
```


2. User is declined

We first repeat the first steps by allowing the user to pick the other activity they have been matched to

SEM / **Matching - pick**

**POST** localhost:8083/matching/pick

Params Authorization Headers (9) **Body** Pre

none form-data x-www-form-urlencoded r

```
1  4
```

**Body** Cookies Headers (11) Test Results

**Pretty** Raw Preview Visualize **Text**

```
1  Application sent
```

Now, we can see that the status of the match is indeed pending

SEM / **Matching - applications**

**GET** localhost:8083/matching/match/PENDING

Params **Authorization** Headers (7) Body Pre-request Script Tests

**Type** **Bearer Token**

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

Heads up! These pa
variables. variables

**Token**

**Body** Cookies Headers (11) Test Results

**Pretty** Raw Preview Visualize **JSON**


```
[
    {
        "matchId": 4,
        "participantId": "d.micloiu@student.tudelft.nl",
        "activityId": 4,
        "ownerId": "luci.tosa11@gmail.com",
        "position": "coach",
        "status": "PENDING"
    }
]
```

The owner does another request now declining the user

SEM / **Matching - decide**

**POST** localhost:8083/matching/decide ...

Params Authorization Headers (9) Body Pre-request Scri

none form-data x-www-form-urlencoded raw binar

```
1 {
2     "matchId" : 4,
3     "decision" : false
4 }
```

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize Text

```
1 Notification sent successfully !
```

As seen above the user gets an email now saying that they have been declined

**New notification regarding rowing competitions**

R rowing.competitions@gmail.com <rowing.competitions@gmail.com> Today at 13:29

To: Diana Micloiu

Unfortunately, you have been denied for activity 4, happening between 2023-03-04T10:12 - 2023-03-04T10:30. We advise you to not give up and try another timeslot or activity.

And the status of the match is updated accordingly to declined

SEM / **Matching - applications**

**GET** localhost:8083/matching/match/DECLINED

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Sett

Type: Bearer Token

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

Heads up! These parameter
variables. variables ↗

Token

Body | Cookies | Headers (11) | Test Results

Pretty | Raw | Preview | Visualize | JSON


```
[
    {
        "matchId": 4,
        "participantId": "d.micloiu@student.tudelft.nl",
        "activityId": 4,
        "ownerId": "luci.tosa11@gmail.com",
        "position": "coach",
        "status": "DECLINED"
    }
]
```

5. **'Activity' microservice** (delete/edit an activity)

SEM / **Activity- edit**

**PATCH** localhost:8084/activities/edit ...

Params Authorization Headers (9) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary GraphQL

```
            "starboard": 0,
            "sculling": 0
        },
        "timeslot": {
            "startTime": "2023-03-04T10:12:00",
            "endTime": "2023-03-04T10:30:00"
        },
        "certificate": "4+",
        "competition": false,
        "gender": "Male",
        "organisation": "Laga"
    }
```

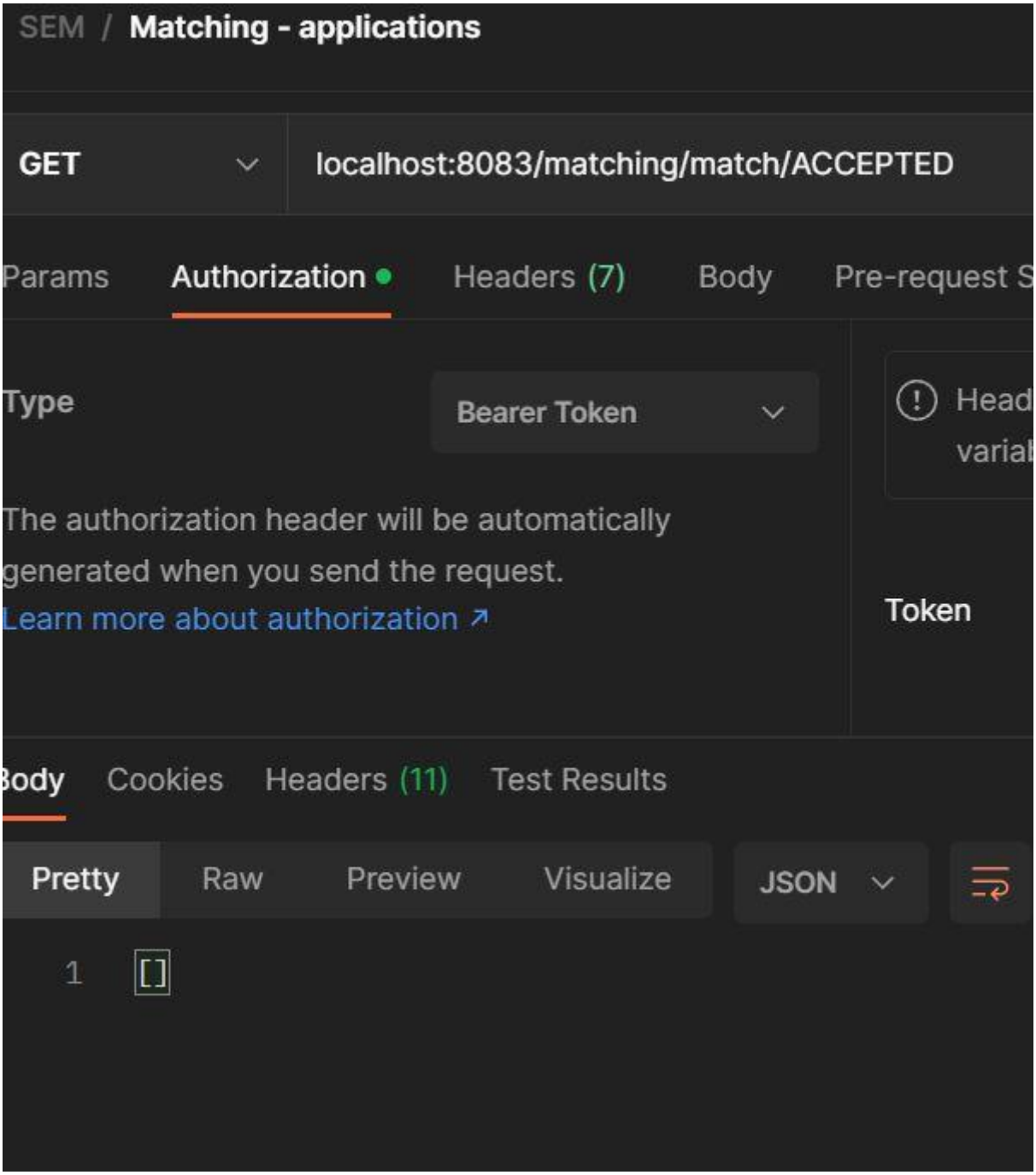
Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "id": 5,
    "ownerId": "luci.tosa11@gmail.com",
    "positions": {
        "cox": 1,
        "coach": 0,
        "port": 0,
        "starboard": 0,
        "sculling": 0
    },
    "timeslot": {
        "startTime": "2023-03-04T10:12:00",
        "endTime": "2023-03-04T10:30:00"
    },
    "certificate": "4+",
    "competition": false,
    "gender": "Male",
    "organisation": "Laga"
}
```

When a user does a request for modifying/deleting an activity a request is done to the **'Matching' microservice** for updating the info accordingly - all the matches to that certain activity are discarded (from the database) and the users accepted previously are announced of the modification (sending a request to the **'Notification' microservice**).

Thus, no more matches are accepted since the activity of the accepted match was modified




And this is the email that the participant gets

**New notification regarding rowing competitions**

R rowing.competitions@gmail.com <rowing.competitions@gmail.com> Today at 13:31

To: Diana Micloiu

Unfortunately, the details for activity 5, happening between 2023-03-04T10:12 - 2023-03-04T10:30have been changed and you have been unenrolled. We advise you to try another timeslot or activity.

6. **'Matching' microservice**

This microservice is also responsible for the automatic addition of the certificates provided in the 'Rowing' scenario but it also has an additional feature of being able to add new certificates that supersede previously added ones. We also included a validation endpoint that verifies if the introduced certificate is among the possible ones used for the current application. (this is also called by other microservices for checking if the certificate provided by the client in the request is an eligible one).

Thus, when running the app certificates "C4" "4+" and "8+" are automatically added in this order. As a consequence, if we try to validate "SEM" we get false, but if we try "C4" we get true

SEM / **Matching - validate certificate**

**POST** localhost:8083/matching/certificate/validate

Params Authorization Headers (9) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary Grap

```
1  SEM
```

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1  false
```

SEM / **Matching - validate certificate**

**POST** localhost:8083/matching/certificate/validate

Params Authorization Headers (9) Body Pre-request Script

none form-data x-www-form-urlencoded raw binary

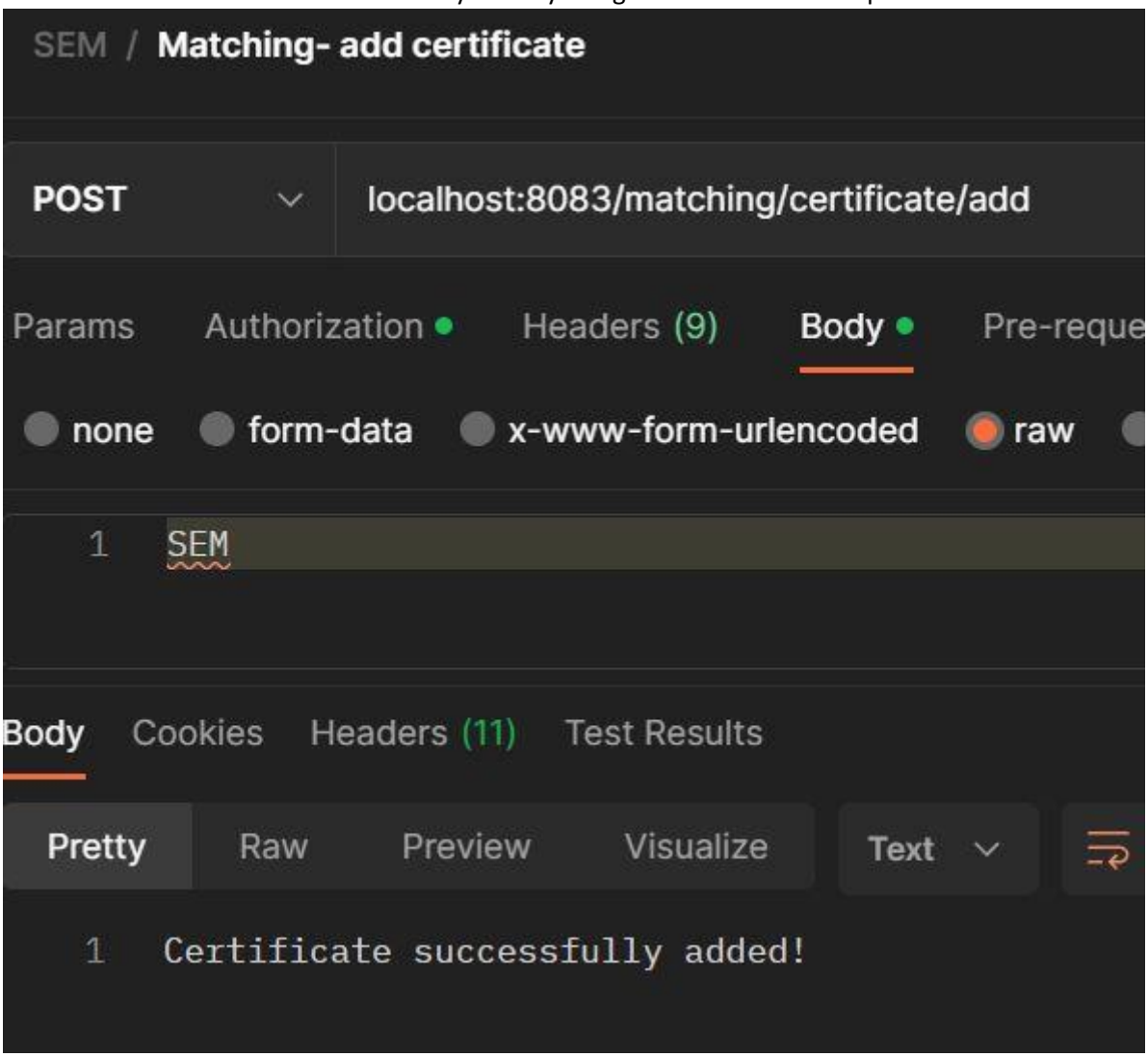
```
1  C4
```

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON

```
1  true
```

Next we can add the certificate to the system by using the addition API endpoint




To wrap up, the **'Matching' microservice** is carefully implemented to take into consideration various special cases and ensure the basic, main functionality of the application whilst also bringing some great addition to the simple required flow of the app.