Semester Project Deadline (DEAD)

Khachidze Tornike 01469313
Ivan Varabyeu 01568715
Hamlet Mkrtchyan 01468003

Bird's-eye view

How does the system as a whole actually work?
Any user can see a list of advertisements without need to be registered or logged in. User can register himself in system to CRUD his own advertisements. Session management is implemented to control, if user is registered and logged in to CRUD.

How do the pieces fit together?
The system has 3 microservices and is build with REST architecture. Each of microservices has its own API with endpoints that are used to communicate.

How MS1, MS2, MS4 are connected?
MS1 is responsible for managing users and session tokens. MS2 is responsible for managing advertisements, while checking internal, if users have enough rights to perform actions (delete advertisements of other users is not allowed). To check if user is logged in and can do CRUD operations with advertisements a request to MS1 is sent to validate the token. The token is generated by MS1 for a user, when he is "logged in" in MS4. MS4 sends a post request with user credentials, MS1 checks them and returns a token in response. Then MS4 can try CRUD operations on advertisement with the API from MS2.

Lessons learned

What were your experiences in the project?
It was interesting to implements the real distributed application. Rest has showed himself as easy to understand and to create architecture.

What were the problems?
One of the drawbacks that we found is MS communication design and changes. Planning an MS architecture needs very good requirements documentation and defined API endpoints already at the start. When someone was changing his backend or type of sended requests, other MS need to be changed also. So testing, support and adding new functionality is a real challenge in MS architecture.

Which problems could be solved, and which couldn't?
our services work together.

Which decisions are your proud of?
At first Tornike has proposed and tested his service with MySQL database, but after discussion in team, we decided to do everything "simple is better". So our concept with in memory repository was born. The drawback of this implementation is that after server restart

all the data is lost. That is why we also have initialization of in memory repository in every MS. It was much easier and faster to start communication between MS.

Which decisions do you regret?
We don't regret - it's experience

If you could start from scratch, what would you do differently?
In the beginning it was challenging for me to construct the structure of my milestone task. I also struggled to connect all the microservices and get them running. For me it was complicated to create user with tokens, but I was able to solve this problem. There were no problems that couldn't be solved, what I am proud of. I think it was a good decision to use memory storage instead of a database, I would do this again in a further project. My program is very simple, and has all the function like login and create users.

Team contribution
We hold some team sessions to discuss possible structural decisions. Each of us has suggested his ideas and at the end we decided to use REST architecture for the project.

gitlab commits

Discussion
How have you eventually deviated from your original plan laid out in SUPD?
As we decided at the beginning - REST remained our main architecture. The only changes that we did, were types of responses and objects to send each other. These things were not so obvious for us.

Interfaces

MS1
Following endpoints are available
api/token/{token} - checks if token is valid
return boolean

MS2
Following endpoints are available:
api/advertisements - list all advertisements
return list of Advertisements
api/advertisements/{id} - show specific advertisement
return Advertisement
api/user{user_id}/advertisements - list all advertisements of logged in user
return of users advertisements
api/create - user creates advertisement
return Http Status
api/update/{id} - user updates advertisement
return Http Status
api/delete/{id} - user deletes advertisement
return Http Status

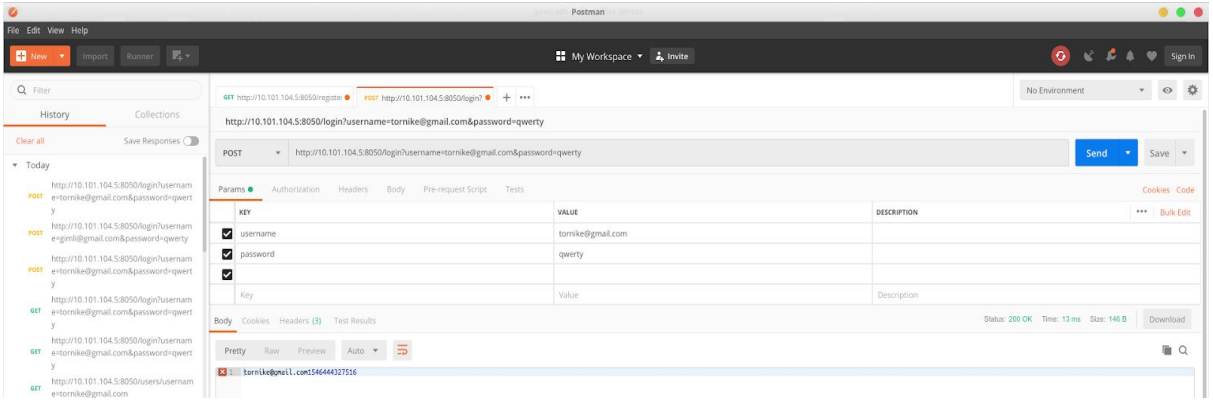
HowTo

How the application is to be launched, initialized and tested?
A “quick start tutorial” that describes how one can use your service.

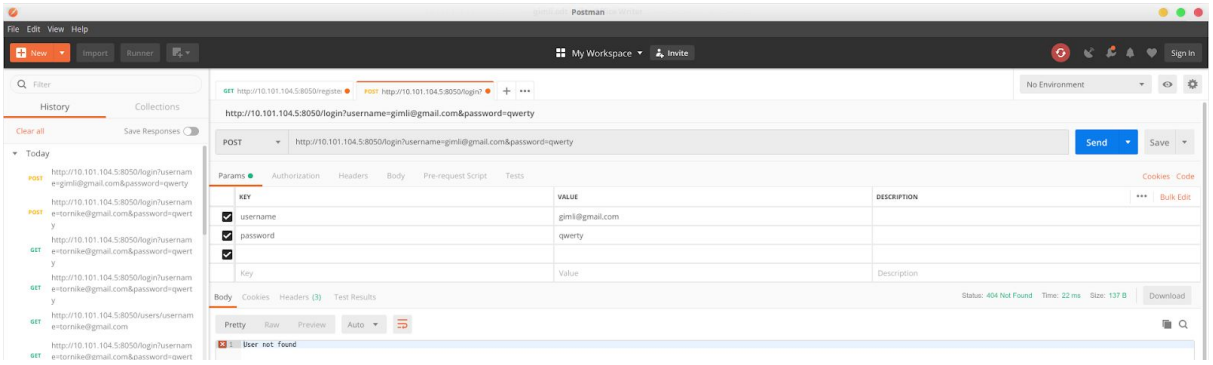
All three MS must either in one subnetwork for test purposes or it is also possible to configure IDE and start MS on different ports. Defining different ports, while packaging in Jar will allow to run MS on one host.

MS1
My Program is programmed with Java and uses Spring Framework REST.
This service implements user registration and user login. For this I use
GET and POST methods.

Log in: for log in I use the POST method, this client sends username and password and checks in
memory storage if the user exists, if yes, the login succeeded, and client gets a token for 1 hour.

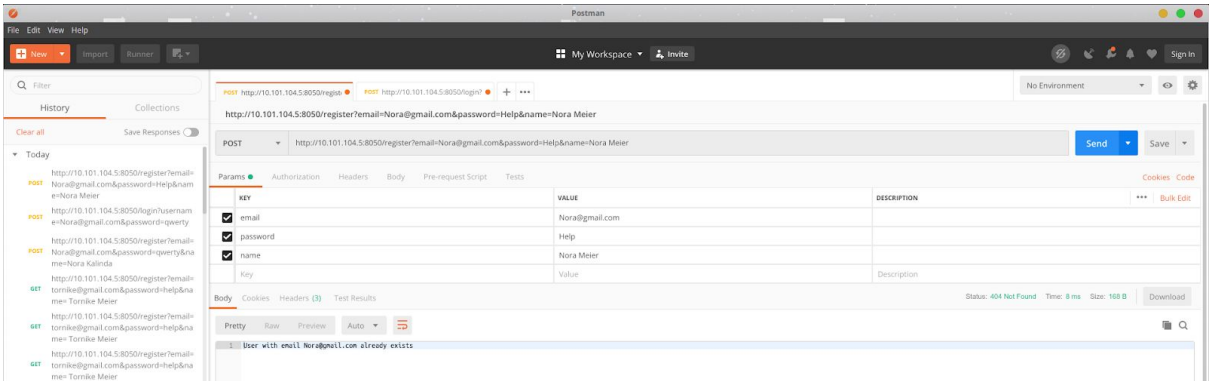


If the user doesn’t exist, user gets an error message “user not found”.

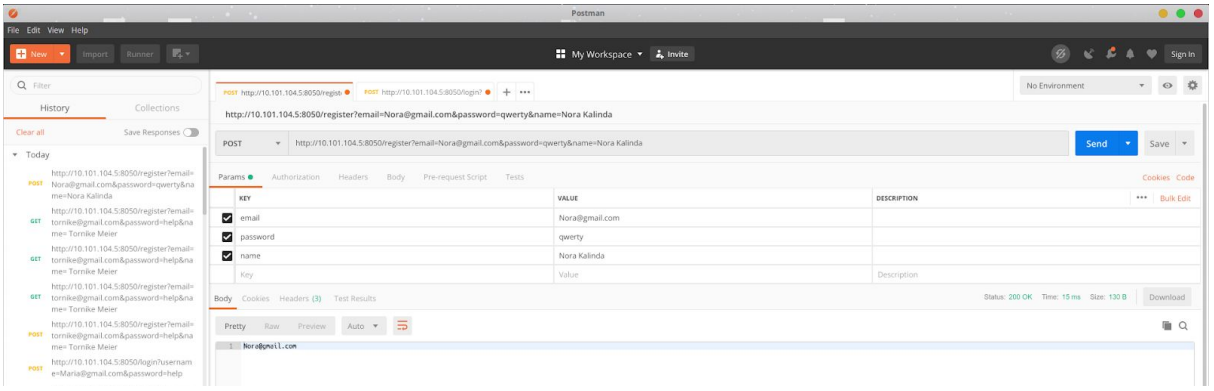


User Registration:

We use GET and POST methods for registration. For registration, we send the username, password and full name. The memory is getting checked if the user is also registrated, the client gets error message “User with email Nora@gmail.com already exists”



If the user doesn’t exist, a new user is added to the memory storage.



MS2

MS is spring boot application and can be packaged in jar and then run on any host. It will be available on localhost:8080. To test it Postman can be used sending GET, POST, PUT, DELETE requests with needed parameters.

MS4

MS4 Takes the responsibilities of a client application. Technically it is divided into a frontend application and a backend api.

Frontend application uses Angular 4 framework with it's Http module to delegate requests to MS4 backend.

MS4 Backend api is defined as an interface to endpoints from ms1 and ms2

/api/deleteAdv/{id}
/api/updateAdv/{id}
/api/createAdv
/api/advertisements
/api/advertisements/{id}
/api/login
/api/token/{token}
/api/register

Initially MS4 was planned to have a heavy client, that gets use of the whole api and leverages the errors from other services and make sure that no error request is sent to other microservices. Nevertheless, the current state remains with error heavy ms4 api, that could possibly delegate faulty data to other microservices.