DSE DEAD

Abdrakhmanova Zh. (01449508), Antonov N. (01348746), Kul H. (01563490), Zhamukhanova A. (01549494)

Team 111

I. Bird's-eye view

MS4 is connected to MS 1-3, it connects to them with URL-Paths in RESTful, so MS4 call this methods with HTTP-method GET, POST, DELETE or PUT. MS3 is connected to MS1 and MS2 also with URL-Paths in RESTful, MS1 and MS2 make the HTTP-Method POST and DELETE to make MS3 database as a copy of MS1 and MS2 databases. So when MS4 ask for a list of advertisements or for notifications, which user can see in his account, MS3 actually searched in his own database and form an actual list of advertisements for search answer and notifications. MS1 and MS2 have their own databases for User and Advertisement Information, respectively.

In prinzip MS1 used by other services to determine whether a user is a known user and currently logged in. When MS4 makes requests to MS2, it gives also userId. At the MS2 part, firstly it would validate if User with such userId exists by making request to MS1. Only if its valid MS2 sends answer to MS4.

II. Lessons learned

- MS 1

My experiences of the project were that I have finally learned and began to understand how REST API is working, how I can connect a Database with Hibernate to a project. The most important point is that I am finally able to understand what a microservice is and what I have to do in order to connect multiple microservice together and how many small programs become one major application. It was certainly a project which expanded my horizon in terms of programming.

The biggest problem I had was to not know where and how i had to start, for a long time I was pretty insecure about the project, I had no point of inspiration or direction. I received a lot of information and read much about different technologies like jwt, oaut, security, rest and decided to use jwt token for

security matters but wasn't really able to fully implement it. The thought of needing a lot of source and coding to accomplish the project was a problem too.
Well, the inspiration problem got solved and I understood that it doesn't take a lot coding to implement everything. I was able to turn a lot of that information I got into knowledge and finally decided to keep everything simple and relay on REST API, Hibernate database and Spring Boot framework. Unfortunately I wasn't able to realize my microservice with JWT and thus the security aspect moved back in my priority list.
I am proud of being able to keep everything very simple and easy understandable.
Honestly I kind of did start from scratch, so I would basically do everything probably the same.

- MS 2

In my implementation I used RESTful technology, it was not the first time, but this semester I have learned how to connect to Database using Spring Hibernate. It have never worked with databases, so it was very informative for me. I find this Spring Hibernate technology very easy to understand and implement. Mainly, I had problems with Spring Annotations (like @Service etc.). I do not have much knowledge about where and how to use them and their meaning. I could not divide my classes in project into separate packages because the project would not compile as all classes were in one package. I think the problem is wrong or absent annotations.

If I could start from scratch I would definitely spend more time for solving the problem above, also I would to look more for security issues, for example what kind of Service is connecting to mine, if he has such rights and so on.

- MS 3

It was my first experience in microservices and RESTful, before i knew only theory about this and about application protocols (HTTP). It was quite hard at the first time to understand the principles of microservices in the field of practice. And it was a big experience especially in learning how to "fit" my microservices to another.

The problems were to understand the principles of Spring-based RESTful good enough to implement it. I rewrote it many times. And to fit it with database exactly was not so simple.

The first time i typed database manually, but then i thought, that it is much better to make it through Hibernate automatically, so tables are fit enough to entities at the start.

I think, that the decision to make the shared database and CRUD repositories is good enough. In this case there is a guarantee that the information is always safe even in case of app failure of failure of the other microservices. And Hibernate technology is very good, simple and effektiv from my point of view.

I find RESTful technology very good for beginners in microservices. Maybe the other patterns are more powerful, but REST is intuitively understandable.

If i could start over, maybe i would try other patterns, cause it would be interesting to compare them in practice. But in my decision i would do all generally the same to my implementation.

The one problem that i couldn't solve are new layers of security, cause it is quite hard for me at this stage do it perfect. But i'm proud of decision to use Hibernate, cause i find it as a very good technology.

- MS 4

For my implementation I had to make Web application for User-Interface and RESTful client to call other three RESTFUL web services. I created Web application using servlet and JSP technology and RESTful Java Client using Jersey. I chose JSP and Servlet for web application because I already had experience with these technology at subject Software Engineering 1. At that time in Software Engineering 1 we were making web-application as team 3 people, but this time I was doing it alone. It helps me to deepen my knowledge in JSP, Servlet technology. It was my first experience in creating some distributed system and RESTful technology. At the beginning it was difficult to understand how our application should work and how communication should be created. After some research of different technologies for Rest client I chose Jersey, because I found it simple and easy to understand. This semester I have learned to create client side of Rest and call microservices. Making MS 4 helped me to see whole picture of communication.

III. Team contribution

Hakan Kul: implementation of MS1 (user registration, authentication and validation).
Aigerim Zhamukhanova: implementation of MS2(means for creating, removing and up- dating advertisements).
Nikita Antonov: implementation of MS3 (search and notifications) and a shared database for users and advertisements in MS3.
Zhanat Abdrakhmanova: implementation of MS4: Web application for User Interface and client side for other three microservices.

IV. Discussion

- MS 1

In SUPD my goal was to implement everything with JWT Token and ensure hence the security aspect and I was using MySQL database but due to lack of talent I decided to implement everything with standard REST API and switched to hibernate database.

- MS 2

In general I have implemented everything what was stated in SUPD, I also added something new. I added more abstraction, namely class User, Manager (where I have implemented needed methods, excluding them from Controller class). The difference from SUPD is in diagram my service had communication only with MS3 and MS4, now it also does requests to MS1. I have also paid attention to our Feedback in SUPD.

- MS 3

I didn't deviate much from my SUPD while my code was already ready and my main task for DEAD was to fit my microservice with other together.

The only two things which i remade were creating the database and notifications class:

In SUPD i typed table structures in SQL manually, but in DEAD i use “create”-mode for Hibernate, cause it makes the exactly tables for entities.

In SUPD i had a separate Notification-structure, but in DEAD i have it as a list of sorted advertisements. In such view it is much better to coordinate my notifications with MS4. I also added boolean method, that proofs, are there new notifications for a registered user or not.

- MS 4

I continued implementing my code in the same way as I started at the beginning. I added client’s methods for all three services, new JSP pages and servlets. The difference from SUPD diagram that Advertisement class doesn’t have userId and userId created in MS1.

V. Interfaces

- MS 1

Nr	URL	Http Method	Mapped to	Description
1	/ms1/register	POST	registerUser	User registration
2	/ms1/login	POST	loginUser	User authentication
3	/ms1/valid/{id}	GET	validate	User validation

- MS 2

Nr.	URL	HTTP method	Mapped to	Description
1	/app/add/advertisement/{userId}	POST	addAdvertisement	create new to advertisement
2	/app/delete/advertisement/{id}	DELETE	deleteAd	Delete existing advertisement by Id
3	/app/advertisements	GET	getAllAds	Get all advertisements

4	/app/getAd/{id}	GET	getAd	Get one existing advertisement by Id
5	/app/userad/{userId}	GET	getUserAd	Get all advertisements belonging to one user (find user by userId)
6	/app/update/{userId}	PUT	updateAd	Update existing Advertisement

- MS 3

URL:

1. **/add/advertisement (HTTP POST)** is used by MS2 to copy or update advertisements in my database accordant to MS2 database
2. **/add/user (HTTP POST)** is used by MS1 to copy or update users in my database accordant to MS1 database
3. **/delete/advertisement/{id} (HTTP DELETE)** is used by MS2 to delete advertisements in my database accordant to MS2 database
4. **/delete/user/{id} (HTTP DELETE)** is used by MS1 to delete users in my database accordant to MS1 database
5. **/search/ (HTTP GET)** can be used to list all existing advertisements
6. **/search/{id}/{advertisement} (HTTP GET)** is used by MS4 for searching in my advertisement database matching the search criteria for registered users (i know a specific user by id)
7. **/search/{advertisement} (HTTP GET)** is used by MS4 for searching in my advertisement database matching the search criteria for unregistered users
8. **/notifications/{id} (HTTP GET)** is used by MS4 to take actual notifications for a specific user(i know this user by id)
9. **/proof/notifications/{id} (HTTP GET)** is used by MS4 to find out, are there actual notifications for a specific user or not(i know this user by id) and take boolean(true/false)

- MS 4

At the start page in nav-bar there are three links to see all products, register and login user. By clicking all Products will be call **Ms2 Nr 3** and user sees all advertisements in database of Ms2. By clicking registration user is redirected to form to fill, than by clicking crete is called **MS1 Nr1** for creating new user. By clicking login is user redirected to form for entering email and password, than by clicking ok is called **MS1 Nr2.** There are also search field. Searching by keyword calls **MS3 Nr7**. When user is logged in System will be appeared new links as "my account" , "add new product to sell" and logout". My account shows all products created by logged user and ables to delete or edit product. By edit is called **MS2 Nr6** and for delete is called **MS2 Nr2.** By clicking "add new product to sell" user is redirected to page to fill necessary fields. By clicking ok is called **MS2 Nr1** for creating new advertisement**.** If user is logged in system and searches advertisement than will be called **Ms3 Nr 6**. The Notification link will be with red spot when there are new Notifications for user. By clicking notifications is called **Ms3 Nr8** and user gets list of advertisements. The "home" link returns user to start page.

VI. HowTo

- MS 1

  Import and run the project in a IDE. The service will start on the IP 10.101.111.13 and Port 8094. Hibernate database will be created and is available on the url http://10.101.111.13:8094/h2-console.

  With Postman you can check with http://10.101.111.13:8094/ms1/register and inserting

  ```
  {
  "id": null,
  "firstName": "Muster",
  "lastName": "Mann",
  "email": "muster@mann.com",
  "password": "1234"
  }
  ```

  Into the body using POST if register if working straight afterward you can check witch http://10.101.111.13:8094/ms1/login if your new registered user is able to authenticate.

- MS 2

Test: The microservice can be tested with the help of MS 1 - 4 or through Postman program.

Quick Start Tutorial:

The Maven should be installed. The project should be imported as maven - spring project.

Download MySQL server and create the mysql data bank on your machine. You can use Command Line for that. After that, the databank url, username and password of your data bank should be written in Application.properties file. It is required to have also the vpn connection in order to run and test the project. The MS2 has IP 10.101.111.17 IP and runs in the port 8080, this information can be changed in file Application.properties. Build and run project as Spring application. Call website http://localhost:8080/ or http://10.101.111.17:8080/ .

- MS 3

Launch and initialization:

Firstly the user of this application should have his own MySQL server. If he doesn't have it, he should go to the official SQL Website, download and tune it. Then he should create his own database for this application and write the application.properties for this database in the application(name, passwort etc). The mode of this database at the first start should be "create", and in that case the application will make the SQL tables for entities automatically.

Then the user should start this application, which is a Spring RESTful service with Maven pom file. If user of this application wants to save this databases even in case of application crash, he should change the mode of this database to "none" after the initialization of the app.

The application will be launched on IP 10.101.111.9 and port 8080, where it is connected to the other parts of the application through VPN.

Testing:

If the other parts of the whole application are already launching, so user can simply run the MS4 in browser, register here and use its interfaces and fields to communicate with MS3.

If user wants to proof the database explizit, or if we don't have MS4, but only MS1 or MS3, so he can use terminal and check the tables of database here.

Finally, if we have only this microservice, and want to test it, it is possible to use Postman application for this. In this application it is possible to make HTTP POST or GET queries and fill accordingly the database or proof methods through this.

- MS 4

The Maven should be installed. Download project and then create war file with maven. After that deploy created war file on Tomcat server. The application will be launched on IP 10.101.111.5 and port 8080.