Distributed Systems Engineering
DEAD - Short Report

Name: Ümit Mordag

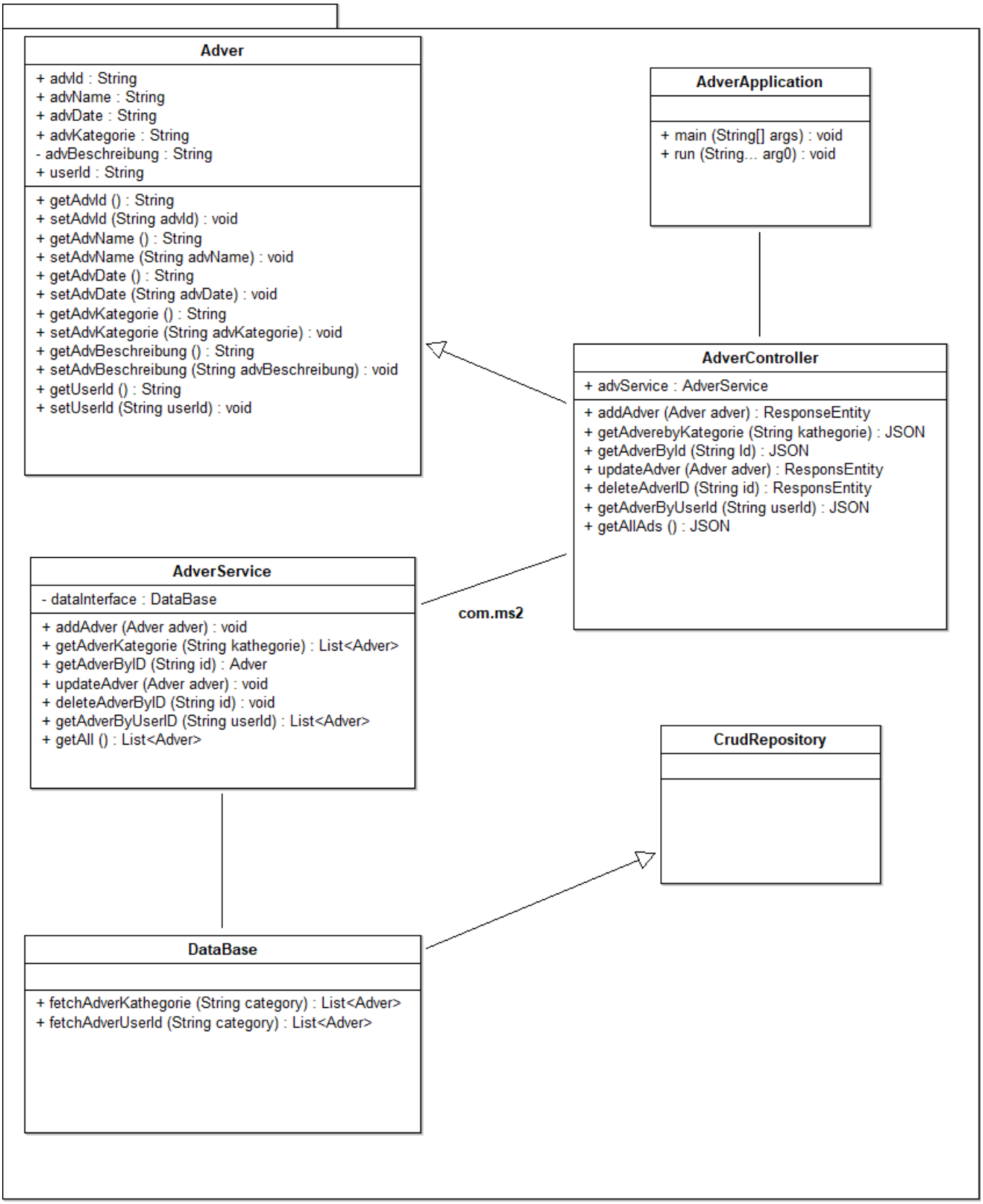
Matriculation Number: 01065738

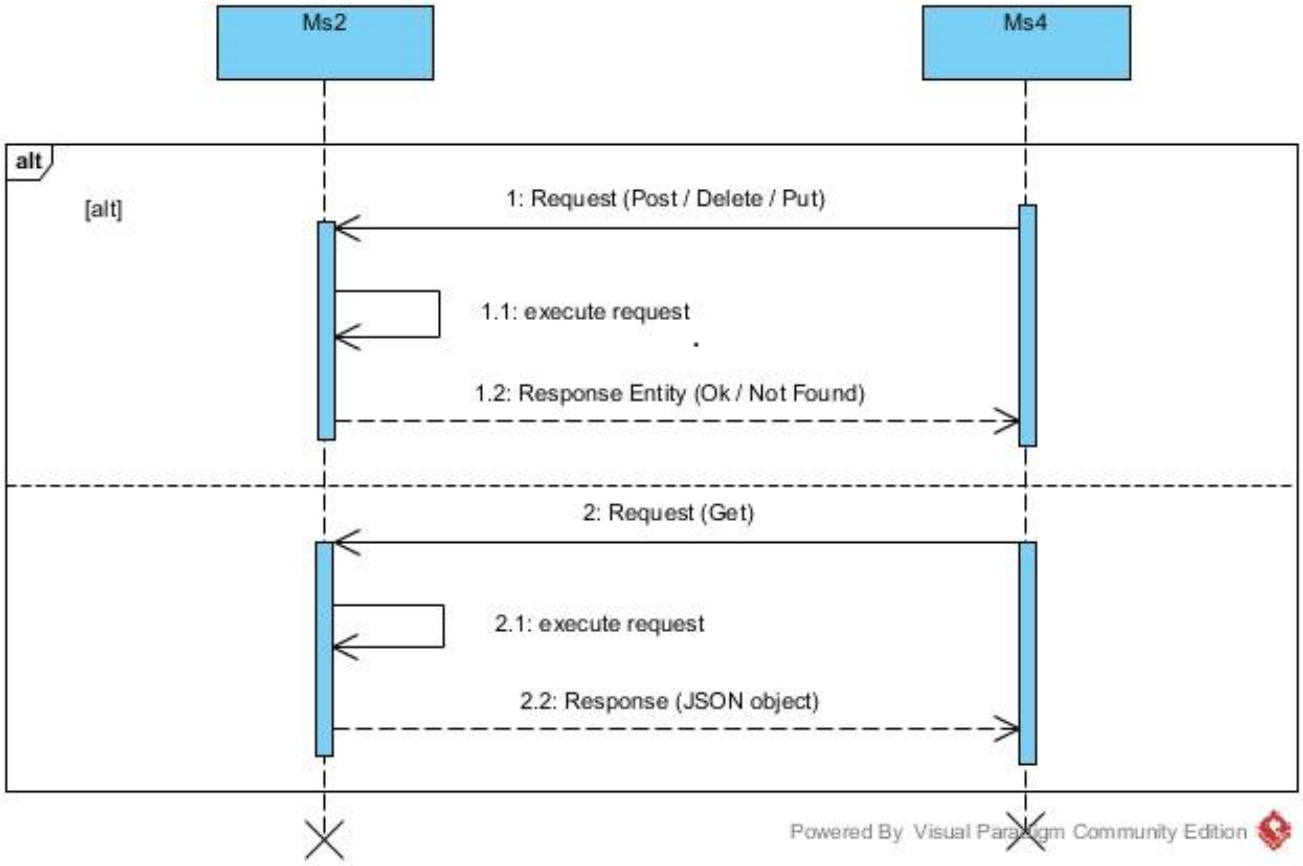
Discussion:

I started with the supd to define the required classes. I first implemented classes in several packages and later I realized that I wouldn´t need that many packages. I have defined the classes "Adver", "AdverController", "AdverService", "DataBase" and as main class "AdverApplication". I don't have to need to save user data in my microservice, because Ms1 has its own user database.

Description of Interface:

I had to implement as mentioned in the Ms2 statement that I provide the Create, Delete, and Update view for users and I worked based on this class. In order to test these functionalities, I used Postman. The Advertisement can be searched in my implementation by Advertisement ID, User ID and category. After user authentication data has been identified with ms1, first request is sent from ms3 to ms2 for help GET request. Of course, I also used Java technologies in my implementation. Some of these are Java Persistence, Apache Derby database. All the advertisements are listed in a database can be displayed by their ID. Each user has different user IDs, so that they cannot see other user`s items. This method guaranties, that users can only manage their own items.


Adver
+ advId : String
+ advName : String
+ advDate : String
+ advKategorie : String
- advBeschreibung : String
+ userId : String
+ getAdvId () : String
+ setAdvId (String advId) : void
+ getAdvName () : String
+ setAdvName (String advName) : void
+ getAdvDate () : String
+ setAdvDate (String advDate) : void
+ getAdvKategorie () : String
+ setAdvKategorie (String advKategorie) : void
+ getAdvBeschreibung () : String
+ setAdvBeschreibung (String advBeschreibung) : void
+ getUserId () : String
+ setUserId (String userId) : void
AdverApplication
+ main (String[] args) : void
+ run (String... arg0) : void
AdverController
+ advService : AdverService
+ addAdver (Adver adver) : ResponseEntity
+ getAdverebyKategorie (String kathegorie) : JSON
+ getAdverById (String Id) : JSON
+ updateAdver (Adver adver) : ResponsEntity
+ deleteAdverID (String id) : ResponsEntity
+ getAdverByUserId (String userId) : JSON
+ getAllAds () : JSON
AdverService
- dataInterface : DataBase
+ addAdver (Adver adver) : void
+ getAdverKategorie (String kathegorie) : List<Adver>
+ getAdverByID (String id) : Adver
+ updateAdver (Adver adver) : void
+ deleteAdverByID (String id) : void
+ getAdverByUserID (String userId) : List<Adver>
+ getAll () : List<Adver>
com.ms2
CrudRepository
DataBase
+ fetchAdverKathegorie (String category) : List<Adver>
+ fetchAdverUserId (String category) : List<Adver>



This sequence diagram shows processing and connection between Ms2 and Ms4. Ms2 waits for a request from Ms4, to process this request and send an appropriate response. For HTTP post, delete and put requests, Ms2 answers either with response entity OK (if request successfully handled) or with response entity NOT FOUND (if request not successfully handled). For HTTP get request, Ms2 answers with JSON object.

Quick start tutorial:

Method name	Http requests	Description	Endpoint
getAdverByKategorie	Get request	This method filters advertisements by category and all of them from database.	http://10.101.108.9:8080/AdvertismentByKat/{kathegorie}
getAdverByUserId	Get request	This method filters advertisements by userId and returns all of them from database.	http://10.101.108.9:8080/AdvertismentByUserId/{userId}
getAdverById	Get request	This method a single advertisement by Id and returns a JSON object.	http://10.101.108.9:8080/AdUpdateById/{id}
getAllAds	Get request	This method returns all advertisement from database	http://10.101.108.9:8080/getAdver
addAdver	Post request	This method saves an advertisement object in database and return response entity.	http://10.101.108.9:8080/Advertisment
deleteAdverID	Delete request	This method deletes a single advertisement by Id and returns response entity.	http://10.101.108.9:8080/AdDeleteById/{id}
updateAdver	Put request	This method updates a single advertisement by ID and returns response entity.	http://10.101.108.9:8080/AdUpdateById/{id}

Sample request and answers for mentioned above methods:

- getAdverByKategorie
  http://10.101.108.9:8080/AdvertismentByKat/Auto


```
[
    {
        "advId": "4",
        "advName": "Volkswagen Golf",
        "advDate": "02.05.2018",
        "advKategorie": "Auto",
        "advBeschreibung": "Wenig KM",
        "userId": "32"
    },
    {
        "advId": "5",
        "advName": "Opel Astra",
        "advDate": "23.07.2018",
        "advKategorie": "Auto",
        "advBeschreibung": "200 PS",
        "userId": "32"
    },
    {
        "advId": "6",
        "advName": "Renault Clio",
        "advDate": "12.05.2018",
        "advKategorie": "Auto",
        "advBeschreibung": "Super Angebot",
        "userId": "56"
    }
]
```


- getAdverByUserId
  http://10.101.108.9:8080/AdvertismentByUserId/32


```
[
    {
        "advId": "4",
        "advName": "Volkswagen Golf",
        "advDate": "02.05.2018",
        "advKategorie": "Auto",
        "advBeschreibung": "Wenig KM",
        "userId": "32"
    },
    {
        "advId": "5",
        "advName": "Opel Astra",
        "advDate": "23.07.2018",
        "advKategorie": "Auto",
        "advBeschreibung": "200 PS",
        "userId": "32"
    }
]
```


- getAdverById
  http://10.101.108.9:8080/AdvertismentById/9


```
{
    "advId": "9",
    "advName": "Esstisch",
    "advDate": "21.07.2018",
    "advKategorie": "Marktplatz",
    "advBeschreibung": "für 6 Personen",
    "userId": "367"
}
```

- getAllAds

http://10.101.108.9:8080/getAdver


```
{
    "advId": "3",
    "advName": "Garconiere in 1010",
    "advDate": "01.01.2017",
    "advKategorie": "Immobilien",
    "advBeschreibung": "Zenral und Luxus",
    "userId": "134"
},
{
    "advId": "4",
    "advName": "Volkswagen Golf",
    "advDate": "02.05.2018",
    "advKategorie": "Auto",
    "advBeschreibung": "Wenig KM",
    "userId": "32"
},
{
    "advId": "5",
    "advName": "Opel Astra",
    "advDate": "23.07.2018",
    "advKategorie": "Auto",
    "advBeschreibung": "200 PS",
    "userId": "32"
},
{
    "advId": "6",
    "advName": "Renault Clio",
    "advDate": "12.05.2018",
    "advKategorie": "Auto",
    "advBeschreibung": "Super Angebot",
    "userId": "56"
},
{
    "advId": "7",
    "advName": "Chefsessel",
    "advDate": "31.05.2018",
    "advKategorie": "Marktplatz",
    "advBeschreibung": "gleich Abholung",
    "userId": "321"
},
```


- addAdver

http://10.101.108.9:8080/Advertisment


```
{
  "advId": "1",
  "advName": "Audi",
  "advDate": "14.02.2017",
  "advKategorie": "Auto",
  "advBeschreibung": "wie neu",
  "userId": "123"
}
```


- deleteAdverID

http://10.101.108.9:8080/AdDeleteById/6

- updateAdver

http://10.101.108.9:8080/AdUpdateById/6


```
{
  "advId": "6",
  "advName": "mercedes benz",
  "advDate": "14.02.2002",
  "advKategorie": "Auto",
  "advBeschreibung": "wenig kilometer, Sitzheizung",
  "userId": "123"
}
```

Service Deployment:

I deployed microservice 2 on my PC with using provided Open VPN server. My server address on port number 8080 is http://10.101.108.9. When I am connected to VPN and execute my implementation, other team colleagues can reach MS2.

Ideas for testing and presentation:

MS2 doesn't crucially depend on the other microservices. Microservice 2 can be tested with sending HTTP request to my endpoint. I tested my implementation with application "POSTMAN". I tested all my methods, functions and achieved to get responses as JSON and response entity. These results are represented in section "Sample request and answers for mentioned above methods".

The current state of my implementation:

My current implementation provides all functionalities, including creating, getting, deleting, updating and filtering by parameter of advertisement objects. Response to other microservices will be in JSON format or response entity. My service runs successfully on provided VPN address. All tasks finished.