Report SUPD

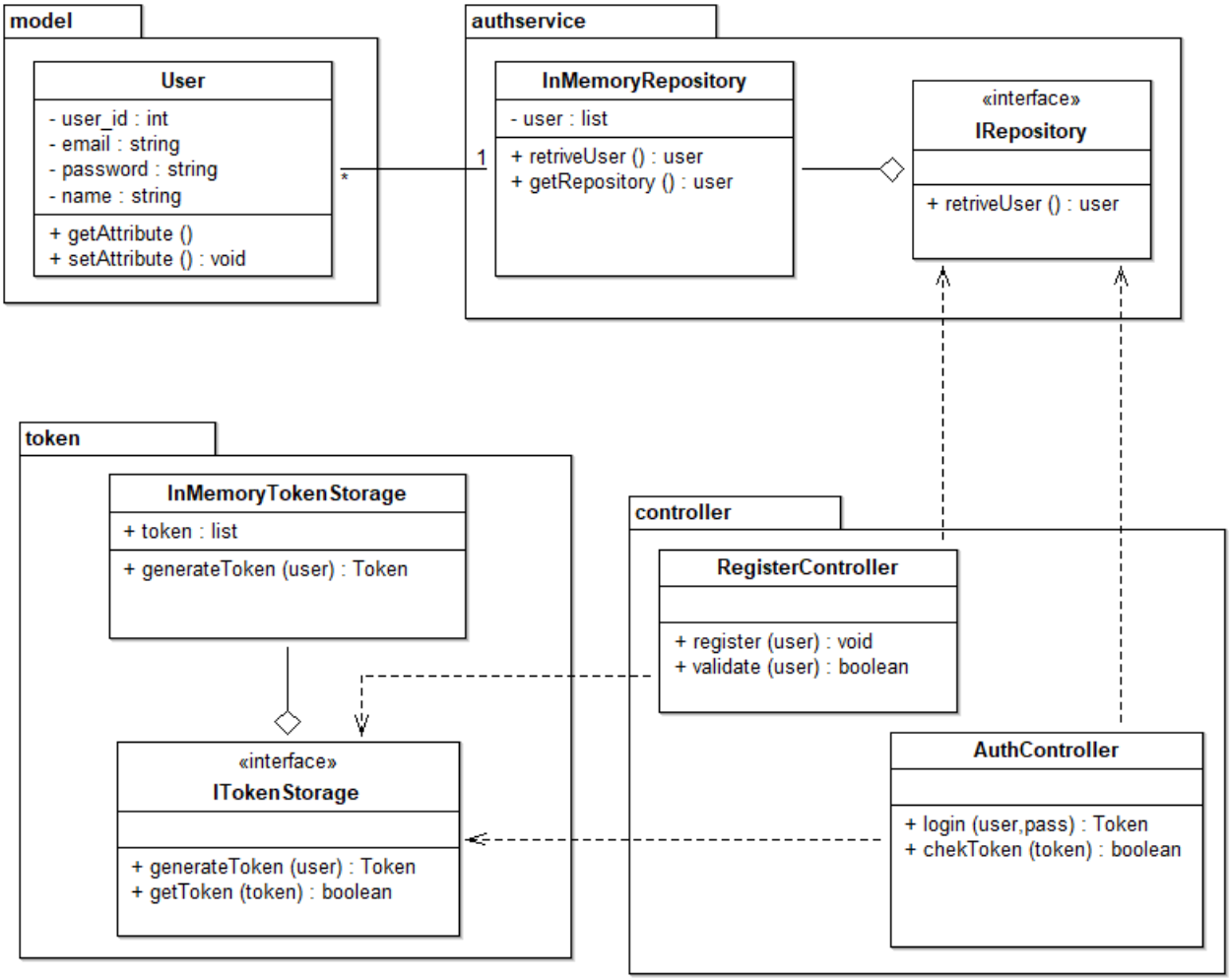
Microservice 1

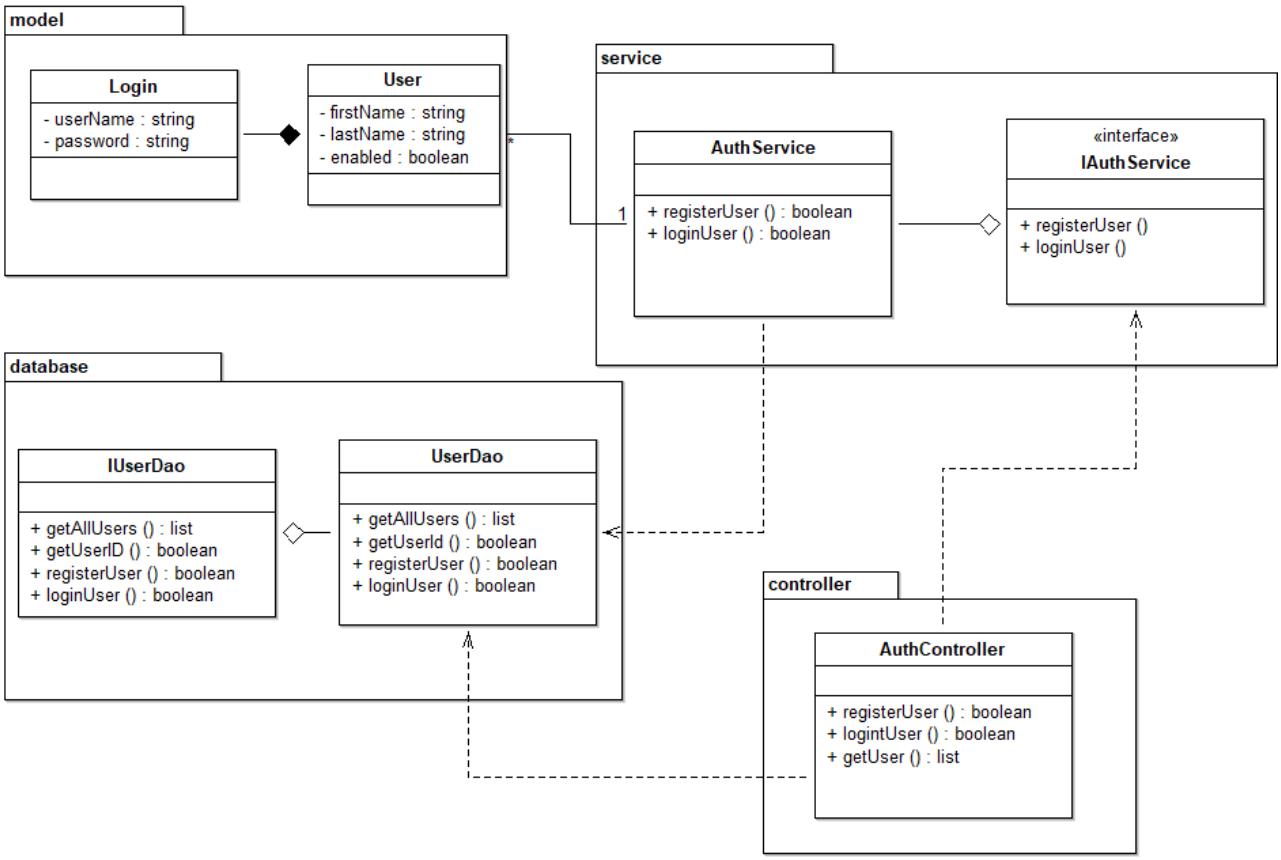
Tornike Khachidze
01469313

Design decisions

Design decesion
For my MS1 I decided to use springboot, a brandnew framework, designed to simplify the bootstrapping and development. I implemented my project in Java and springboot simplifies the use of Java, and creates many details on its own. Springboot is very easy to use.
I decided not to use a database, we use memory storage instead. We are checking someone's data (email and password) and then we should get back a Json file, for example {"email":ivan@gmail.com, "password": 1234}
The external interface has many classes: user repository class, retrieverUser (checks username and password), return User. I have token storage, that generates tokens for users, also method get token. When to ken exists, return true, if not: false. I have two controllers, one auth controller, that checks token and logs in users. Second controller is the registration controll, method registers users and validate users. Also user validator, that validates email and password.

The difference between the two approaches is, that one works with databases, the other uses memory storage.


model
User
- user_id : int
- email : string
- password : string
- name : string
+ getAttribute ()
+ setAttribute () : void
*
1
authservice
InMemoryRepository
- user : list
+ retriveUser () : user
+ getRepository () : user
«interface»
IRepository
+ retriveUser () : user
token
InMemoryTokenStorage
+ token : list
+ generateToken (user) : Token
«interface»
ITokenStorage
+ generateToken (user) : Token
+ getToken (token) : boolean
controller
RegisterController
+ register (user) : void
+ validate (user) : boolean
AuthController
+ login (user,pass) : Token
+ chekToken (token) : boolean



Deployment

The service is currently running on my computer, with local host.
Following paths are working, when service is started:

http://10.101.104.5:8050/login?username=ivan@gmail.com&password=qwerty

Testing and presentation

For testing my implementation I used postman and VPN connect, I also used Get and Post methods and I can get Json String back.

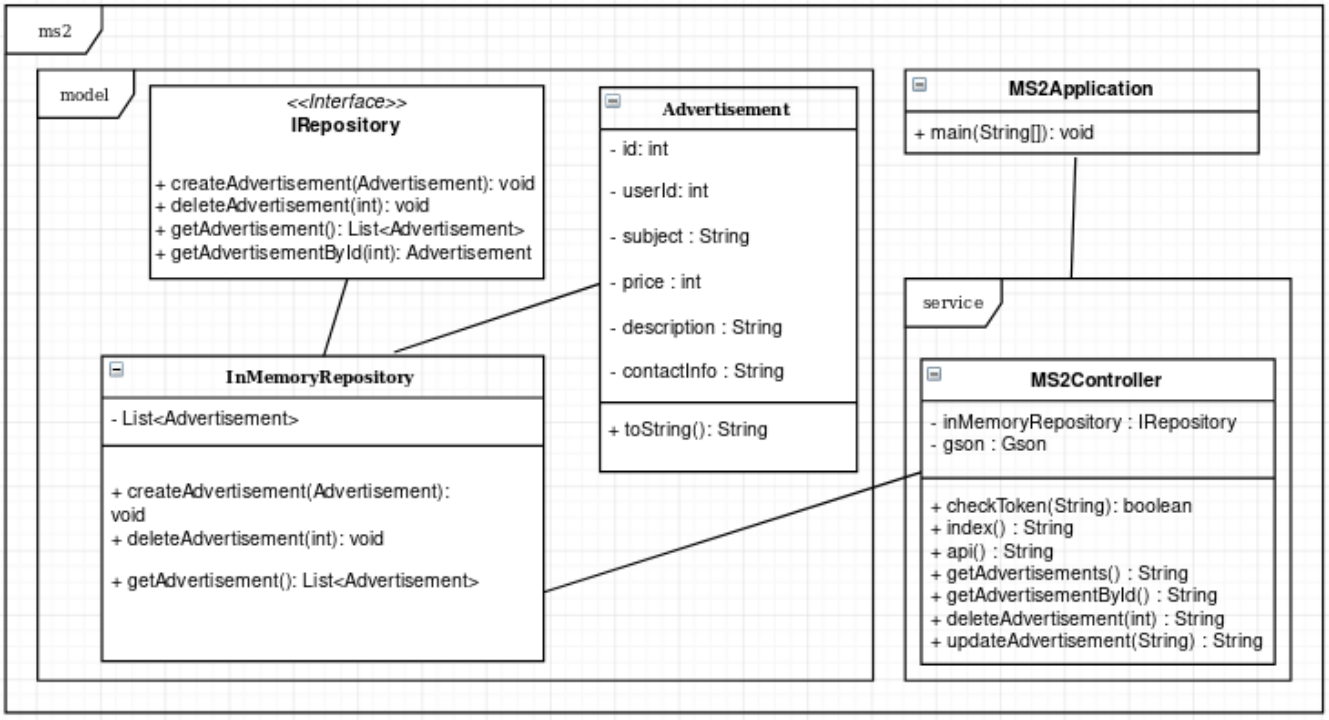
Status

For testing I created some classes, that were checked in list, if they exist, and returned a string with the full name.
Some things are still missing, for example in AuthController I need to implement Check tokens, and token storage, that generates tokens for users. Also missing Registration Controller for user registration and validate.

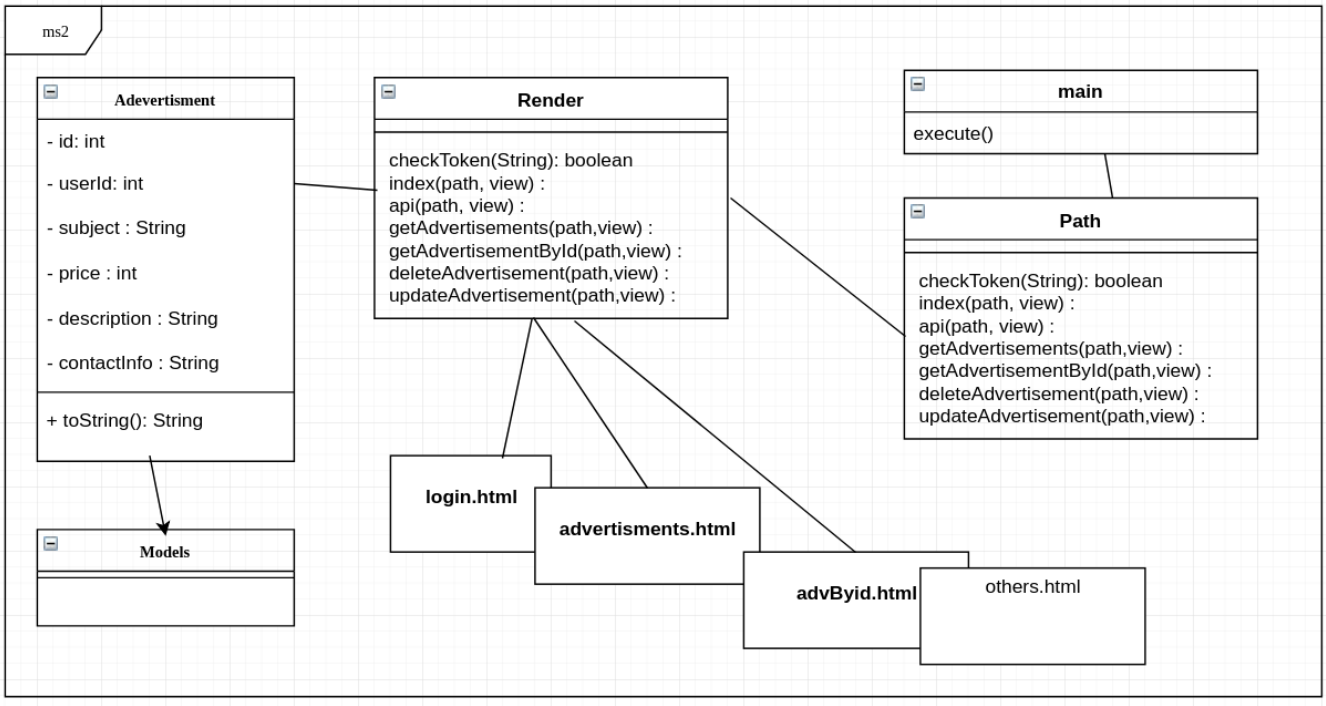
Our team is connected via VPN and everything we implemented already is working, but its still in the testing phase.

Microservice 2

Design decisions



Picture 1 – UML_1 (current implementation)



Picture 2 – UML_2 (possible implementation)

The second UML diagram shows another possible sollution using Django Framework and Python language with server side rendering of frontend interface. It will also use Sqlite dabase to save data of model definitions. Models are written with the help of built in objects-relational mapper, that will transform them in proper sql commands.

As we learn mostly Java I decided to use java for the implementation of ms2. Spring Boot web project with maven is used. REST API architecture is used to communicate with other services. Information from ms2 will be send in JSON format. In case of object store I decided for in-memory- store. Each reboot of the server will delete all the data. Other possible solution were using DB integration or Java serialization.

With request (create, update, delete, show all for current user) from the client ms2 will check the session token sent with the request, by sending request to authentication service. If everything is OK, the client will get the JSON data. Request of all available advertisements and one will not need any token.

ms2 will be available on following paths:

api/advertisements

api/advertisements/{id}

api/{user_id}/advertisements

api/{user_id}/advertisements/{id}

api/{user_id}/create

api/{user_id}/update/{id}

api/{user_id}/delete/{id}

Deployment

As we are using VPN all our services will be started on personal laptops with exposed ports to listen requests. In my ms2 it will be 8080 port. VPN connection will be needed every time to override possible firewall and proxy problems. Server is available on 10.101.104.13:8080.

Testing and presentation

Current test were done manually without any constraints or JUnit tests. Coming soon… For test purposes Postman was used to send simple GET requests. On presentation the hole project will run in VPN network with working requests and responses between microservices.

Status

Git workflow accepted and learned by all teammates.

Data model is done without constraints.

Creating objects is hardcoded for test purposes.

Response is done in JSON format.

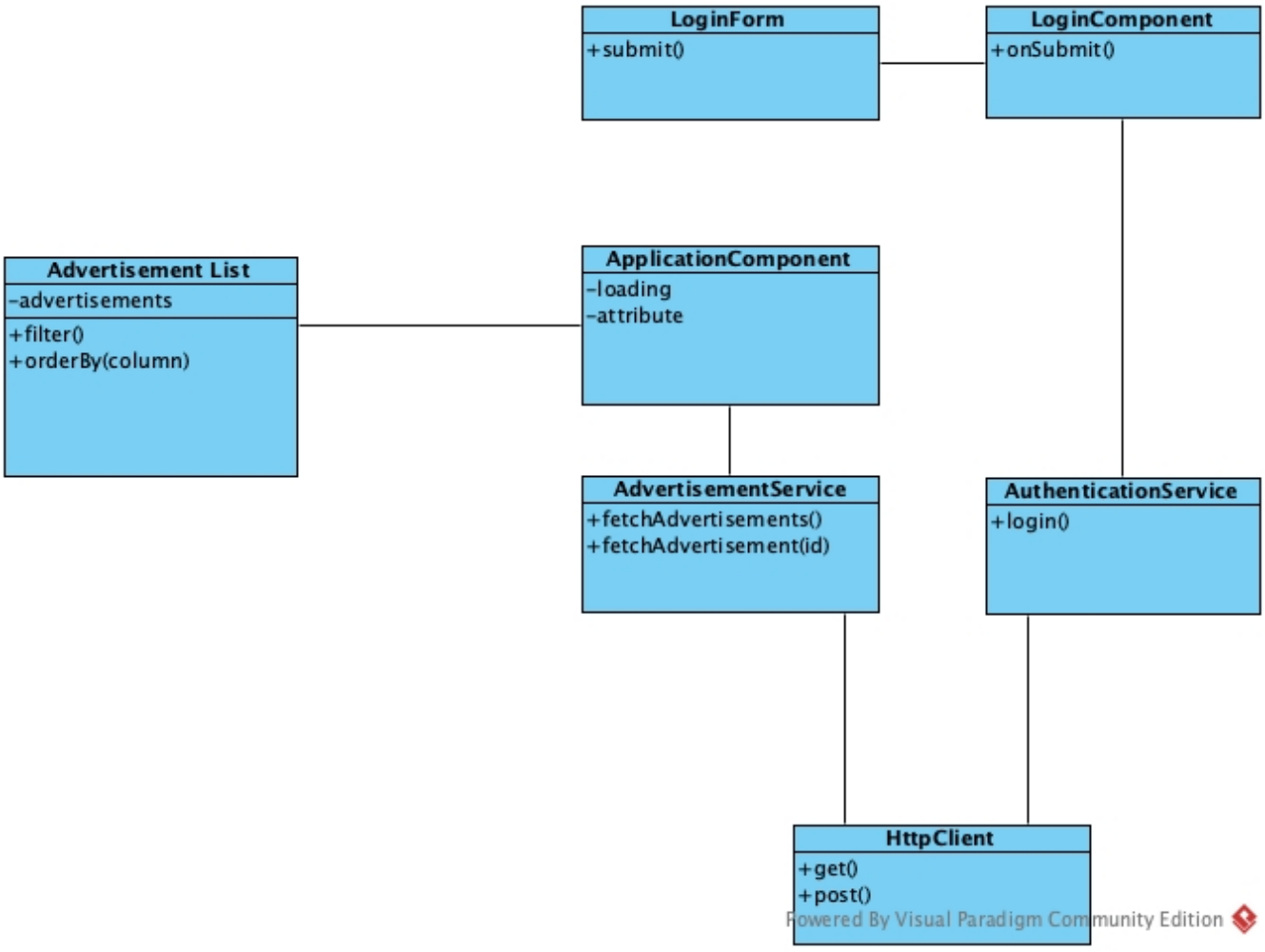
Connection between services and simple requests are working already.

Following paths are working, when service is started:

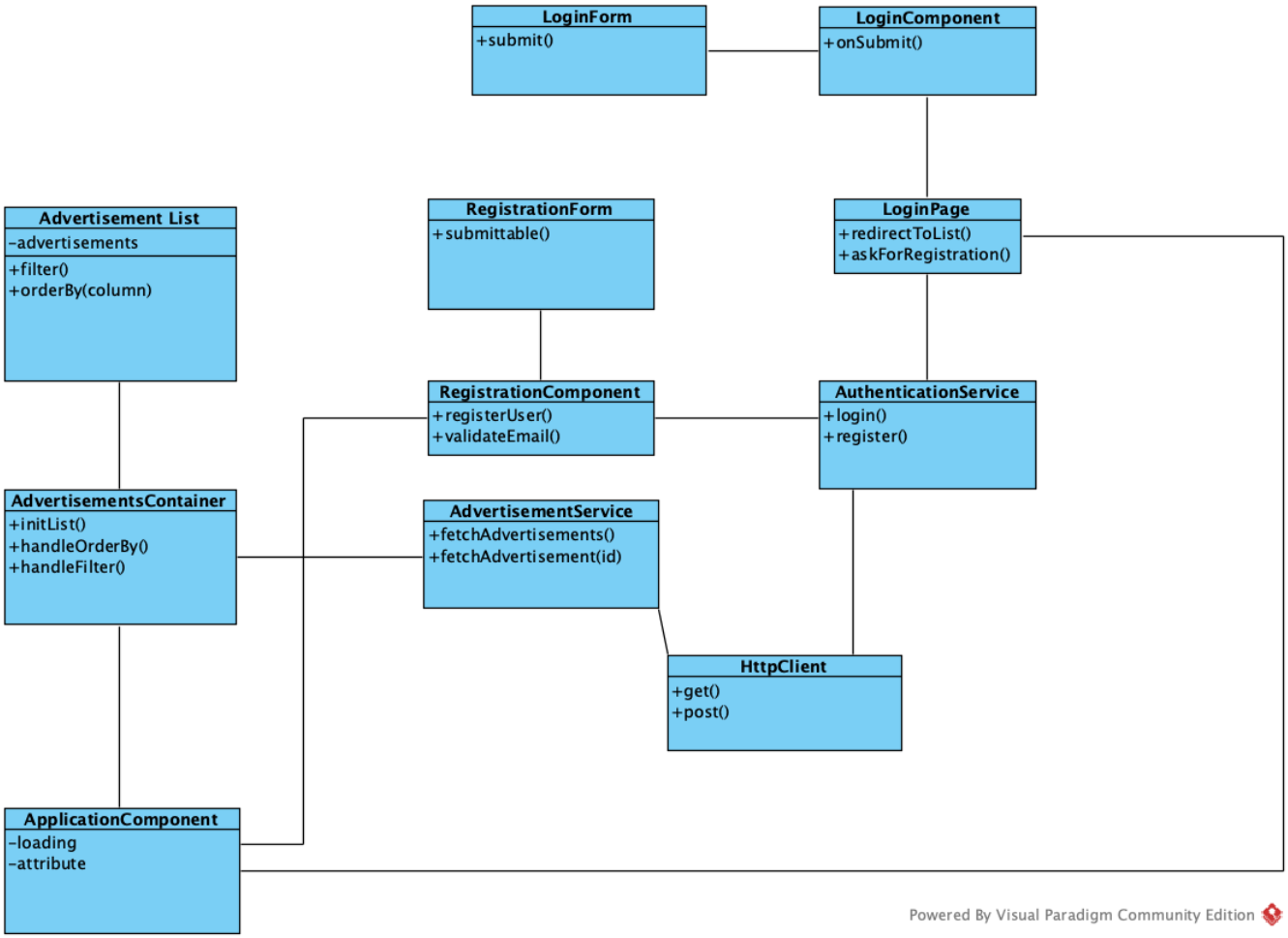
10.101.104.13:8080/api/advertisements
10.101.104.13:8080/api/advertisements/{id}

TODO: make all paths working, work with token, request data validation, unit tests.

Microservice 4



Version 1



Version 2

Design decisions

Angular 4 javascript framework was choosen as a main tool. Huge benefit of it is the component structure and typescript language. Client runs in a container – presenter fashion, while keeping containers “intelligent” and presenters as dumb as possible.
ApplicationComponent acts as a root component. It is actually where the whole client is integrated. It directly communicates to AdvertisementService that provides an interface to correctly fetch and Observe results from ms2.
After successfully fetching the data, ApplicationComponent binds it to AdvertisementList which renders a table of advertisements.
Current choice fell onto the first design version, since it has more monolithic pattern, which corresponds to the simpler needs of a client.

Deployment

Project could be built via angular cli tool with ng build command. It can be deployed to the remote server as a web application. Alternatively for development purposes a dev server could be used.

Testing and presentation

Status

List of advertisement and the login are implemented. Users are able to authenticate them and proceed to the advertisements list.
Next steps are

- More customizable advertisements with a richer data.
- Integrate add and delete advertisements via REST endpoints of MS2.
- Search functionality to filter the right content out.
- Styling.

API specification

Show Advertisements

Returns json data about all available advertisements

- **URL** /api/advertisements
- **Method:** `GET`
- **URL Params** None
- **Data Params** None
- **Success Response:**
  - **Code:** 200
    Content: [{"id":0,"userId":2,"subject":"first","price":50,"description":"computer nice","contactInfo":"tel. 473823748"},{"id":1,"userId":2,"subject":"second","price":50,"description":"computer nice","contactInfo":"tel. 473823748"}]
  -
- **Error Response:**

  Coming soon...

Show Advertisement by ID

Returns json data about one specific advertisement

- **URL** /api/advertisements/{id}
- **Method:** `GET`
- **URL Params**

  Required: id=[integer]
- **Data Params** None
- **Success Response:**
  - **Code:** 200
    Content: {"id":1,"userId":2,"subject":"second","price":50,"description":"computer nice","contactInfo":"tel. 473823748"}
- **Error Response:**

  Coming soon...

Show user advertisements

Coming soon...

Show user advertisement by ID

Coming soon...

Create Advertisement

Coming soon...

Update Advertisement by ID

Coming soon...

Delete Advertisement by ID

Coming soon...