Deviation From "Status Update"

There were no major deviations from Design Decisions laid out in the Status-Update-Report. The Java Application Framework Dropwizard ist still the foundation of ms1. Eventually ms2 was reimplemented using the same framework. The following libraries – orchestrated by Dropwizard – are used to provide a RESTful web service:

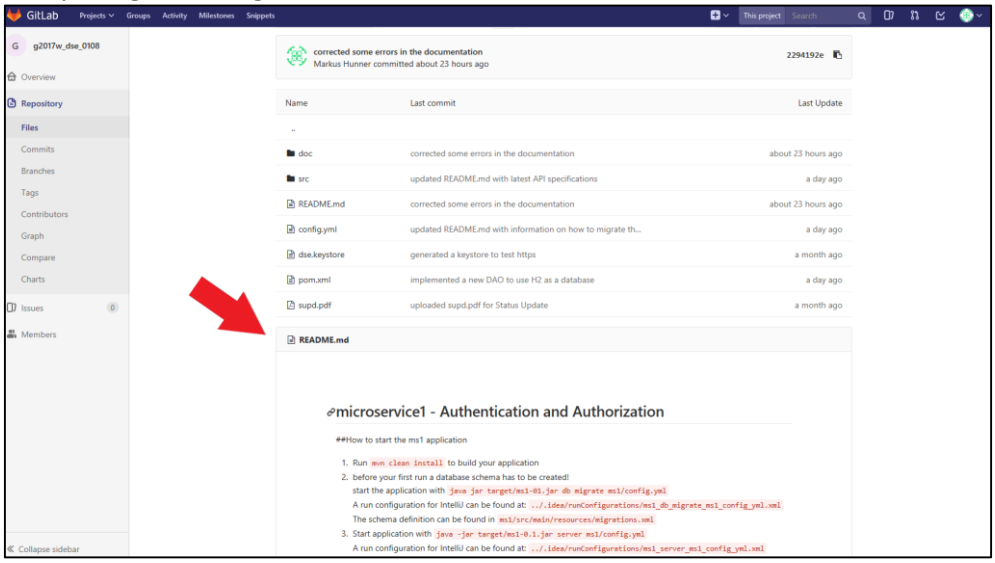
- Jetty for HTTP serving.
- Jersey for REST modeling.
- Jackson for JSON parsing and generating'.
- Logback for logging.

For deployment the "Status Update" mentioned providing the microservices as DOCKER containers to allow easy testing of a distributed system on a single machine. This idea was rejected as the other team members didn't feel comfortable using another and unknown deployment method than the provided VPN.

During development I wrote a documentation on how to run, configure and use my microservice as a help for the other team members. Most topics asked for in this report are already covered by this documentation. It can be found in the Gitlab-repository at:

/MS1/README.MD

It can be easily navigated using the Gitlab webinterface:

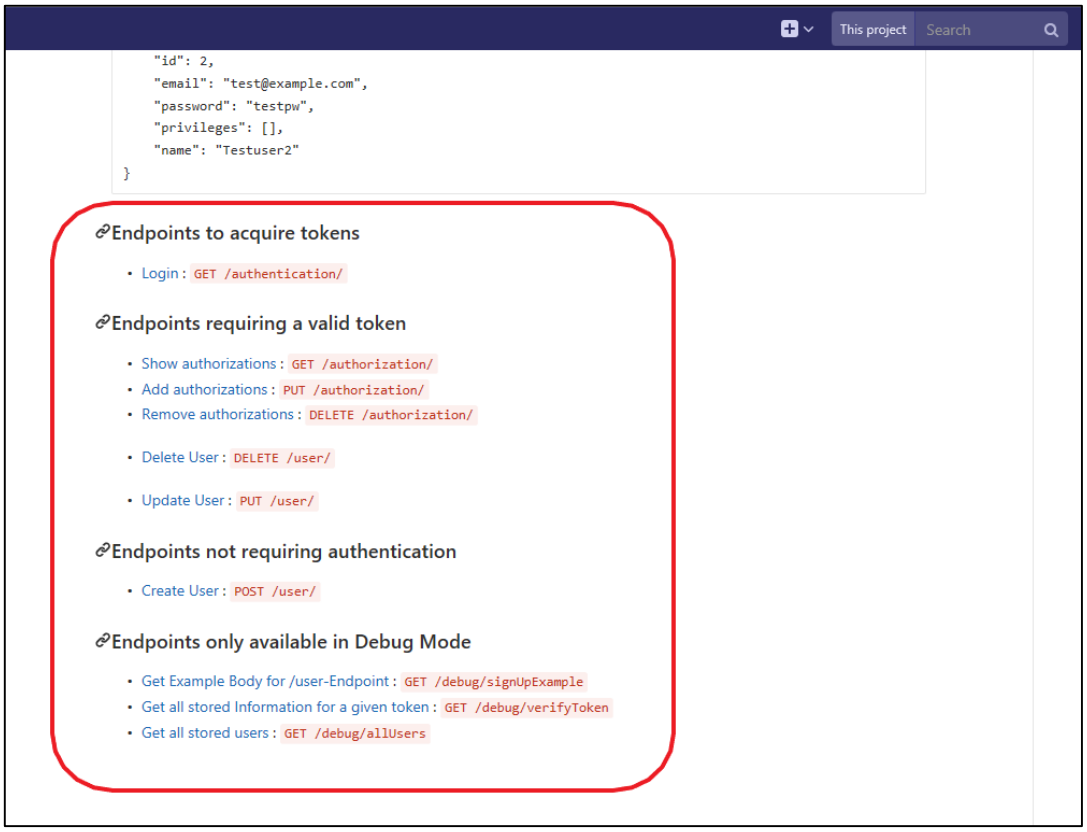


As outlined as a future goal in the "Status Update" ms1 now uses the H2 SQL Database via hibernate to store user data and the IDs of the ToDoLists they are authorized to see. Notable features still include the support of unecrypted JsonWebTokens und fully encrypted JsonWebTokens. The Service supports HTTP and HTTPs via a generated keystore-File. For easier debugging and for the purpose of presenting the ms1 special debugging API-Endpoints were implemented. Those endpoints are only available if the application is run in "DebugMode"
Additional information about using those features can be found in the configuration file "config.yml".

Additionally to the documentation in "ms1/README.md" a jdoc generated documentation can be found at */ms1/doc/javadocs/index.html*

API Endpoints:

A very detailed documentation of this REST-API, detailing needed authentication, Body contents, example responses and possible error responses can be found at the end of "ms1/README.md":



It can easily be navigated using the Gitlab webinterface by clicking on the provided links. The links lead to the corresponding files at "ms1/doc/…".

/AUTHENTICATE/ supporting GET-requests using HTTP Basic Authentication[1] and if successful serving a session token for the authenticated user

Ideally the orchestrating microservice (ms3) would generate a special Login-JWT, filled with the claims "Username" and "Password" signed and encrypted with a Private key used by ms3. Ms1 could then use the corresponding Public key of ms3 to parse those claims and return a valid Bearer-/Session-Token. Unfortunately the team member assigned to ms3 left the group. Therefore for the sake of simplicity the HTTP-Standard of "Basic Authentication" is used for the initial login – this also means that username and password have to be send unencrypted if using the service without HTTPS.

/AUTHENTICATE/REFRESH supporting GET-requests with a JWT Bearer Token, returning a new Token with a TTL of 45 minutes. This can be used to refresh a token.

/USER/ supporting POST-requests to create new user accounts, PUT to update a user's password or email and DELETE to delete a user's account.

/AUTHORIZE/ supporting PUT, POST, DEL to manage a user's privileges

[1] RFC 2617

Only available if application is run in debug mode (see "config.yml")

/DEBUG/ALLUSERS supporting GET-requests and returns all database entries in JSON-format.

/DEBUG/VERIFYTOKEN supporting GET-requests and returns the database entry for the provided token. This data can then be compared to the data stored in the token via www.jwt.io

/DEBUG/SIGNUPEXAMPLE supporting GET-requests and returns a JSON-Object as used in the POST- and PUT-Requests to the endpoints /user/

Quick Start Tutorial

Everything needed to configure and run ms1 can be found in the file "ms1/README.md"

1. Run *mvn clean install* to build your application
2. before your first run a database schema has to be created!
   start the application with *java jar target/ms1-01.jar db migrate ms1/config.yml*

A run configuration for IntelliJ can be found at:
../.idea/runConfigurations/ms1_db_migrate_ms1_config_yml.xml

The schema definition can be found in *ms1/src/main/resources/migrations.xml*

3. Start application with *java -jar target/ms1-0.1.jar server ms1/config.yml*

A run configuration for IntelliJ can be found at:
../.idea/runConfigurations/ms1_server_ms1_config_yml.xml

To check that the application is running enter URL *http://localhost:8081* - that's the admin panel

Alternatively you can look for these lines in the console output to find the right URL:

```
INFO  [2018-01-08 19:00:14,413] org.eclipse.jetty.setuid.SetUIDListener: Opened
application@5a39e554{HTTP/1.1,[http/1.1]}{0.0.0.0:8080}

INFO  [2018-01-08 19:00:14,413] org.eclipse.jetty.setuid.SetUIDListener: Opened
application@333d44f6{SSL,[ssl, http/1.1]}{0.0.0.0:8085}

INFO  [2018-01-08 19:00:14,413] org.eclipse.jetty.setuid.SetUIDListener: Opened
admin@350bbd5d{HTTP/1.1,[http/1.1]}{0.0.0.0:8081}
```

Be aware that the running ports have to be configured via the config.yml.
Normally Port 8080 for HTTP and Port 8085 for HTTPS.

Check *config.yml* for details! There are several other configuration parameters. Especially important are the ones listed under *dse2017*.