BILAG 2B

BESKRIVELSE AF SNITFLADE TIL IDENTITY MANAGER

.

1. Generelt

Som nævnt i Bialg 2, Kravspecifikationen har Kunden pt. et Identity Manager-system i drift. Nedenstående er den tekniske dokumentation på dette.

2. Initializing the Database

To initialize the database first create the user and schema by running this script:

mysql -uroot -p < etc/mysql/create_schema_and_user.sql

The tables will be created and migrated (if needed) upon starting the application using Flyway. The migration files are located in the src/main/resources/db/migration/ folder.

3. Launching the application

1. To launch the application using the spring boot maven plugin:

   mvn spring-boot:run

2. You can also launch the application by executing the main method in IdMApplicationConfiguration using your IDE.
3. Finally you can build a distributable jar file run it:
   1. Build the jar

   mvn clean install

   Produces a distributable target/idm-1.0.0-SNAPSHOT.jar file.

   2. Run the jar

   java -jar idm-1.0.0-SNAPSHOT.jar

4. Overriding Configuration Defaults

To override the embedded default configuration in src/main/resources/application.properties you can create a file named application.properties and either place it in the same directory as the jar file, or in a subdirectory named 'config'. If the system can be run using different configurations, it is possible to load files that are specific to a given environment using the spring.profiles.active parameter:

java -jar target/idm-1.0.0-SNAPSHOT.jar --spring.profiles.active="prod"

The command above will load and merge the property files application.properties and application-prod.properties.

It is also possible to specify new values for individual properties using the command line e.g. --propertyKey=propertyValue. The default port for the embedded Tomcat is 8080, controlled by the server.port property. Here is an example of overriding the default port when launching the application:

java -jar target/idm-1.0.0-SNAPSHOT.jar --server.port=9000

For further details about configuring the application see the "Externalized Configuration" section in the Spring Boot reference.

5. API Endpoints and Authentication

There are two API endpoints

https://<server:port>/api/verification

and

http://<server:port>/api/admin

Note: The two endpoints use different schemes! The verification api is available only over https, whilst the admin api is only available over http.

Both endpoints are secured using Http Basic Auth.

The username/password for accessing /api/verification is controlled by the 'idm.verifierUser' and 'idm.verifierPass' properties, the default value is 'verifier'/'verifierpass'.

The username/password for accessing /api/admin is controlled by the 'idm.adminUser' and 'idm.adminPass' properties, the default value is 'admin'/'adminpass'.

6. API - Register New User / Reset existing users password

This endpoint is used to create a new user, or if a user with that username already exists reset the password and the failed_attempts counter as well as set 'enabled' to true.
If the call succeeds 200 OK is returned, otherwise 403 Forbidden is returned.

This endpoint requires two parameters 'user' and 'pw' - and optionally a 'customerId'

http://<server:port>/api/admin/register

Test using curl:

curl -i -X POST --data "user=test1" --data "pw=test" --user admin:adminpass http://localhost:8080/api/admin/register

Verify that data in the database are correct:

mysql -uidm -pidm idm -e "SELECT user,failed_attempts, enabled FROM idm.users WHERE user='test1'"

7. API - Disable a user

This endpoint can be used to disable a user account immediately - i.e. prevent the user from being able to login, until they use the /api/admin/register call to reset the password, and enabled property. If the call succeeds 200 OK is returned, otherwise 403 Forbidden is returned.

This endpoint requires one parameters 'user' - and optionally a 'customerId'

http://<server:port>/api/admin/disable

Test using curl:

curl -i -X POST --data "user=test1" --user admin:adminpass http://localhost:8080/api/admin/disable

Verify that data in the database are correct:

mysql -uidm -pidm idm -e "SELECT user,failed_attempts, enabled FROM idm.users WHERE user='test1'"

8. API - Verification

This endpoint can be used for checking if the user, password and customerId combination is correct. If they are, 200 OK is returned (and the users 'failed*attempts' field is set to 0), otherwise 403 Forbidden is returned and the 'failed*attempts' field is incremented. When the users failed*attempts count exceeds the value specified in the property 'idm.failedAttemptsLimit' the user will be disabled and subsequent verify calls with correct username/password will still give 403 Forbidden. When this happens the user has to re-register with the same username - which will set a new password, and reset the failed*attempts count and re-enable the user account.

This endpoint requires two parameters 'user' and 'pw' - and optionally a 'customerId' This endpoint can only be accessed over HTTPS

curl -kL -i -X POST --data "user=test1" --data "pw=test" --user verifier:verifierpass https://localhost:8443/api/verification/verify

Verify that data in the database are correct:

mysql -uidm -pidm idm -e "SELECT user,failed_attempts, enabled FROM idm.users WHERE user='test1'"