From: **Wolfgang Karl** wolfgang.karl@dnetwo.de
Subject: OAUTH2
Date: 31. July 2018 at 19:39
To: n.mueller@safeplace.de

KW

Hallo Normen,

sorry für die verspätete Rückmeldung, bin aktuell leider (wie zuvor telefonisch erwähnt) ordentlich im Stress.

Bin bedauerlicherweise nur über meinen mobile Hotspot drinnen (sitze gerade im Zug auf dem Weg nach Lausanne), weshalb ich das OAUTH2-Skeletton über folgenden Link bereitstelle:

https://zvn.io/s/oauth2.zip

Kurz zu den Endpoints, entsprechend der OAUTH2.md vom Mai (Anhang) lassen sich diese über untenstehende Requests aufrufen. Sämtliche DTO's für die Requests und die entsprechenden Antworten wurden erstellt.

Register:

curl -H "Content-Type: application/json" -X POST -d '{
"type": "pull",
"client_name": "test-app",
"client_url": "localhost:9000",
"client_description": "my test application",
"redirect_url": "localhost:8080/myService/redirectUrlAfterLogin"
}' http://localhost:9000/api/oauth2/register

Auth:

curl -G 'http://localhost:9000/api/oauth2/auth' --data-urlencode 'client_id=abcd' --data-urlencode 'redirect_uri=http://localhost:9000/endpoint' --data-urlencode 'response_type=code'

Token:

curl -H "Content-Type: application/json" -X POST -d '{
"grant_type": "authorization_code",
"client_id": "123456789",
"redirect_uri": "http://localhost:9000/endpoint",
"client_secret": "1a2b3c4d5e",
"code": "123456789"
}' http://localhost:9000/api/oauth2/token

Refresh Token:

curl -G 'http://localhost:9000/api/oauth2/refresh_token' --data-urlencode 'access_token=1a2b3c4d5f6g7h'

Verify Token Validity:

curl -G 'http://localhost:9000/api/oauth2/verify_token_validity' --data-urlencode 'access_token=1a2b3c4d5f6g7h'

Query Endpoint:

curl -G 'http://localhost:9000/api/oauth2/query_endpoint' --data-urlencode 'access_token=1a2b3c4d5f6g7h' --data-urlencode 'endpoint=http://localhost/api/users'

data-urlencode "endpoint=http://localhost/api/users"

Falls diese Snippets im Mail nicht korrekt dargestellt werden sollten, finden sich diese als Kommentar über dem jeweiligen Endpoint im **OauthService.java** der API.

TODOS:

- Anbindung Data Source
- Lagom Commands (geht zwar auch ohne, jedoch kann man so z.B. keine Integration Tests für POST-Requests erstellen).
- *Optional:* Eventuell Verschmelzung mit folgendem OAUTH Provider (dependency ist bereits drinnen): https://github.com/nulab/scala-oauth2-provider

---

Betreffend der Deployments auf AWS haben wir dort Docker Repos für die unterschiedlichen Microservices angelegt, zu finden in der ECS: https://console.aws.amazon.com/ecs/home . Grundlegend kann man das jeweilige docker repo einfach einbinden und neue Images pushen, woraufhin automatisch eine neue (aktualisierte) Instanz erstellt + hochgefahren wird und im Anschluss die alte heruntergefahren + zerstört (terminated). Es müsste hierzu aber noch eine schönere Kubernetes-Lösung geben, nämlich über die EKS: https://aws.amazon.com/eks/

Müsste ich mir bei Gelegenheit aber noch genauer ansehen.

Viele Grüsse aus der Schweiz,

Wolfgang

OAUTH2 Microservice.md